

intel®

**MCS-80/85™  
FAMILY  
USER'S  
MANUAL**

OCTOBER 1979

**\$7.50**

Intel Corporation, 1979  
121506-001

**MOS**

**80**

**MOS**

**85**







# MCS-80/85™ FAMILY USER'S MANUAL

OCTOBER 1979

The following are trademarks of Intel Corporation and may be used only to describe Intel products: Intel, Insite, Inteltec, Library Manager, Megachassis, Micromap, Multibus, PROMPT, RMX/80, UPI, Intelevison,  $\mu$ Scope, Promware, MCS, ICE, iSBC, BXP, iCS, and the combination of MCS, ICE, iSBC or iCS with a numerical suffix.

Intel Corporation Assumes No Responsibility for the Use of Any Circuitry Other Than Circuitry Embodied in an Intel Product. No Other Circuit Patent Licenses are Implied.

# Table of Contents

## CHAPTER 1

Part 1: Introduction to the Functions of a Computer .....	1-1
Part 2: Introduction to MCS-85™ .....	1-6

## CHAPTER 2

Functional Description .....	2-1
------------------------------	-----

## CHAPTER 3

System Operation and Interfacing .....	3-1
--	-----

## CHAPTER 4

The 8080 Central Processor Unit .....	4-1
---------------------------------------	-----

## CHAPTER 5

The Instruction Set .....	5-1
Instruction Set Index .....	5-19

## CHAPTER 6

### DEVICE SPECIFICATIONS

#### MCS-85

8085A/8085A-2 8-Bit Microprocessor .....	6-1
8155/8156/8155-2/8156-2 RAM/IO/Counter-Timer .....	6-17
8185/8185-2 RAM .....	6-32
8355/8355-2 ROM/IO .....	6-37
8755A/8755A-2 EPROM/IO .....	6-45

#### MCS-80

8080A/8080A-1/8080A-2 8-Bit Microprocessor .....	6-56
8224 Clock Generator and Driver .....	6-64
8228/8238 System Controller and Bus Driver .....	6-68
8801 Clock Generator Crystal .....	6-72

#### System Support Components

8205 High-Speed 1 Out of 8 Binary Decoder .....	6-74
8212 8-Bit Input/Output Port .....	6-80
8218/8219 Bi-polar Microcomputer Bus Controllers .....	6-90
8237/8237-2 High Performance Programmable DMA Controller .....	6-101
8257/8257-5 Programmable DMA Controller .....	6-115
8259A/8259A-2/8259A-8 Programmable Interrupt Controller .....	6-132
8282/8283 Octal Latch .....	6-150
8286/8287 Octal Bus Transceiver .....	6-154



# Table of Contents (Continued)

## Peripherals\*

8041A/8641A/8741A Universal Peripheral Interface .....	6-158
8202 Dynamic RAM Controller .....	6-159
8251A Programmable Communication Interface .....	6-160
8253/8253-5 Programmable Interval Timer .....	6-161
8255A/8255A-5 Programmable Peripheral Interface .....	6-162
8271 Programmable Floppy Disk Controller .....	6-163
8272 Single/Double Density Floppy Disk Controller .....	6-164
8273 Programmable HDLC/SDLC Protocol Controller .....	6-165
8275 Programmable CRT Controller .....	6-166
8279/8279-5 Programmable Keyboard/Display Interface .....	6-167
8291 GPIB Talker/Listener .....	6-168
8292 GPIB Controller .....	6-169
8293 GPIB Transceiver .....	6-170
8294 Data Encryption Unit .....	6-171
8295 Dot Matrix Printer Controller .....	6-172

## Static RAMs

2114A 1024 x 4 Bit Static RAM .....	6-173
2141 4096 x 1 Bit Static RAM .....	6-174
2142 1024 x 4 Bit Static RAM .....	6-175
2148 1024 x 4 Bit Static RAM .....	6-176

## ROMs/EPROMs

2716 16K (2K x 8) UV Erasable PROM .....	6-177
2732 32K (4K x 8) UV Erasable PROM .....	6-178
2758 8K (1K x 8) UV Erasable Low Power PROM .....	6-182
3604A, 3624A 4K (512 x 8) High-Speed PROM .....	6-183
3605A, 3625A 4K (1K x 4) PROM .....	6-184
3628 8K (1K x 8) Bipolar PROM .....	6-187
3636 16K (2K x 8) Bipolar PROM .....	6-190

## CHAPTER 7

### DEVELOPMENT AIDS

ICE-80™ MCS-80™ In-Circuit Emulator .....	7-1
UPP-103 Universal PROM Programmer .....	7-7
Intellec® Microcomputer Development System Model 220 .....	7-9
Intellec® Microcomputer Development System Model 230 .....	7-13
ISIS-II Diskette Operating System .....	7-17
PL/M-80 High Level Programming Language Inteltec® Resident Computer .....	7-20
ICE-85™ MCS-85™ In-Circuit Emulator .....	7-23
SDK-85 MCS-85™ System Design Kit .....	7-27
Fortran-80 8080/8085 ANS Fortran 77 Inteltec® Resident Compiler .....	7-33
Basic-80 Extended ANS 1978 Basic Inteltec® Resident Interpreter .....	7-37
ICIS-Cobol™ Software Package .....	7-40

## APPENDIX

Applications of MCS-85™ .....	A1-1
-------------------------------	------

\*This section contains partial data sheets. For complete details, refer to the Intel Peripheral Design Handbook.

# Chapter 1

## Introduction

MOS

CMOS

MOS

CMOS



# Chapter 1 Introduction

# CHAPTER 1

## PART 1: INTRODUCTION TO THE FUNCTIONS OF A COMPUTER

This chapter introduces certain basic computer concepts. It provides background information and definitions which will be useful in later chapters of this manual. Those already familiar with computers may skip this material, at their option.

### A TYPICAL COMPUTER SYSTEM

A typical digital computer consists of:

- a) A central processor unit (CPU)
- b) A memory
- c) Input/output (I/O) ports

The memory serves as a place to store **Instructions**, the coded pieces of information that direct the activities of the CPU, and **Data**, the coded pieces of information that are processed by the CPU. A group of logically related instructions stored in memory is referred to as a **Program**. The CPU "reads" each instruction from memory in a logically determined sequence, and uses it to initiate processing actions. If the program sequence is coherent and logical, processing the program will produce intelligible and useful results.

The memory is also used to store the data to be manipulated, as well as the instructions that direct that manipulation. The program must be organized such that the CPU does not read a non-instruction word when it expects to see an instruction. The CPU can rapidly access any data stored in memory; but often the memory is not large enough to store the entire data bank required for a particular application. The problem can be resolved by providing the computer with one or more **Input Ports**. The CPU can address these ports and input the data contained there. The addition of input ports enables the computer to receive information from external equipment (such as a paper tape reader or floppy disk) at high rates of speed and in large volumes.

A computer also requires one or more **Output Ports** that permit the CPU to communicate the result of its processing to the outside world. The output may go to a display, for use by a human operator, to a peripheral device that produces "hard-copy," such as a line-printer, to a

peripheral storage device, such as a floppy disk unit, or the output may constitute process control signals that direct the operations of another system, such as an automated assembly line. Like input ports, output ports are addressable. The input and output ports together permit the processor to communicate with the outside world.

The CPU unifies the system. It controls the functions performed by the other components. The CPU must be able to fetch instructions from memory, decode their binary contents and execute them. It must also be able to reference memory and I/O ports as necessary in the execution of instructions. In addition, the CPU should be able to recognize and respond to certain external control signals, such as INTERRUPT and WAIT requests. The functional units within a CPU that enable it to perform these functions are described below.

### THE ARCHITECTURE OF A CPU

A typical central processor unit (CPU) consists of the following interconnected functional units:

- Registers
- Arithmetic/Logic Unit (ALU)
- Control Circuitry

Registers are temporary storage units within the CPU. Some registers, such as the program counter and instruction register, have dedicated uses. Other registers, such as the accumulator, are for more general purpose use.

#### Accumulator:

The accumulator usually stores one of the operands to be manipulated by the ALU. A typical instruction might direct the ALU to add the contents of some other register to the contents of the accumulator and store the result in the accumulator itself. In general, the accumulator is both a source (operand) and a destination (result) register.

Often a CPU will include a number of additional general purpose registers that can be used to store operands or intermediate data. The availability of general purpose



registers eliminates the need to "shuffle" intermediate results back and forth between memory and the accumulator, thus improving processing speed and efficiency.

### Program Counter (Jumps, Subroutines and the Stack):

The instructions that make up a program are stored in the system's memory. The central processor references the contents of memory, in order to determine what action is appropriate. This means that the processor must know which location contains the next instruction.

Each of the locations in memory is numbered, to distinguish it from all other locations in memory. The number which identifies a memory location is called its **Address**.

The processor maintains a counter which contains the address of the next program instruction. This register is called the **Program Counter**. The processor updates the program counter by adding "1" to the counter each time it fetches an instruction, so that the program counter is always current (pointing to the next instruction).

The programmer therefore stores his instructions in numerically adjacent addresses, so that the lower addresses contain the first instructions to be executed and the higher addresses contain later instructions. The only time the programmer may violate this sequential rule is when an instruction in one section of memory is a **Jump** instruction to another section of memory.

A jump instruction contains the address of the instruction which is to follow it. The next instruction may be stored in any memory location, as long as the programmed jump specifies the correct address. During the execution of a jump instruction, the processor replaces the contents of its program counter with the address embodied in the Jump. Thus, the logical continuity of the program is maintained.

A special kind of program jump occurs when the stored program "**Calls**" a subroutine. In this kind of jump, the processor is required to "remember" the contents of the program counter at the time that the jump occurs. This enables the processor to resume execution of the main program when it is finished with the last instruction of the subroutine.

A **Subroutine** is a program within a program. Usually it is a general-purpose set of instructions that must be executed repeatedly in the course of a main program. Routines which calculate the square, the sine, or the logarithm of a program variable are good examples of functions often written as subroutines. Other examples might be programs designed for inputting or outputting data to a particular peripheral device.

The processor has a special way of handling subroutines, in order to insure an orderly return to the main program. When the processor receives a **Call** instruction, it increments the Program Counter and stores the counter's contents in a reserved memory area known as the **Stack**. The Stack thus saves the address of the instruction to be executed after the subroutine is completed. Then the pro-

cessor loads the address specified in the **Call** into its Program Counter. The next instruction fetched will therefore be the first step of the subroutine.

The last instruction in any subroutine is a **Return**. Such an instruction need specify no address. When the processor fetches a **Return** instruction, it simply replaces the current contents of the Program Counter with the address on the top of the stack. This causes the processor to resume execution of the calling program at the point immediately following the original **Call** instruction.

Subroutines are often **Nested**; that is, one subroutine will sometimes call a second subroutine. The second may call a third, and so on. This is perfectly acceptable, as long as the processor has enough capacity to store the necessary return addresses, and the logical provision for doing so. In other words, the maximum depth of nesting is determined by the depth of the stack itself. If the stack has space for storing three return addresses, then three levels of subroutines may be accommodated.

Processors have different ways of maintaining stacks. Some have facilities for the storage of return addresses built into the processor itself. Other processors use a reserved area of external memory as the stack and simply maintain a **Pointer** register which contains the address of the most recent stack entry. The external stack allows virtually unlimited subroutine nesting. In addition, if the processor provides instructions that cause the contents of the accumulator and other general purpose registers to be "pushed" onto the stack or "popped" off the stack via the address stored in the stack pointer, multi-level interrupt processing (described later in this chapter) is possible. The status of the processor (i.e., the contents of all the registers) can be saved in the stack when an interrupt is accepted and then restored after the interrupt has been serviced. This ability to save the processor's status at any given time is possible even if an interrupt service routine, itself, is interrupted.

### Instruction Register and Decoder:

Every computer has a **Word Length** that is characteristic of that machine. A computer's word length is usually determined by the size of its internal storage elements and interconnecting paths (referred to as **Busses**); for example, a computer whose registers and busses can store and transfer 8 bits of information has a characteristic word length of 8-bits and is referred to as an 8-bit parallel processor. An eight-bit parallel processor generally finds it most efficient to deal with eight-bit binary fields, and the memory associated with such a processor is therefore organized to store eight bits in each addressable memory location. Data and instructions are stored in memory as eight-bit binary numbers, or as numbers that are integral multiples of eight bits: 16 bits, 24 bits, and so on. This characteristic eight-bit field is often referred to as a **Byte**.

Each operation that the processor can perform is identified by a unique byte of data known as an **Instruction**

**Code or Operation Code.** An eight-bit word used as an instruction code can distinguish between 256 alternative actions, more than adequate for most processors.

The processor fetches an instruction in two distinct operations. First, the processor transmits the address in its Program Counter to the memory. Then the memory returns the addressed byte to the processor. The CPU stores this instruction byte in a register known as the **Instruction Register**, and uses it to direct activities during the remainder of the instruction execution.

The mechanism by which the processor translates an instruction code into specific processing actions requires more elaboration than we can here afford. The concept, however, should be intuitively clear to any logic designer. The eight bits stored in the instruction register can be decoded and used to selectively activate one of a number of output lines, in this case up to 256 lines. Each line represents a set of activities associated with execution of a particular instruction code. The enabled line can be combined with selected timing pulses, to develop electrical signals that can then be used to initiate specific actions. This translation of code into action is performed by the **Instruction Decoder** and by the associated control circuitry.

An eight-bit instruction code is often sufficient to specify a particular processing action. There are times, however, when execution of the instruction requires more information than eight bits can convey.

One example of this is when the instruction references a memory location. The basic instruction code identifies the operation to be performed, but cannot specify the object address as well. In a case like this, a two- or three-byte instruction must be used. Successive instruction bytes are stored in sequentially adjacent memory locations, and the processor performs two or three fetches in succession to obtain the full instruction. The first byte retrieved from memory is placed in the processor's instruction register, and subsequent bytes are placed in temporary storage; the processor then proceeds with the execution phase. Such an instruction is referred to as **Variable Length**.

#### Address Register(s):

A CPU may use a register or register-pair to hold the address of a memory location that is to be accessed for data. If the address register is **Programmable**, (i.e., if there are instructions that allow the programmer to alter the contents of the register) the program can "build" an address in the address register prior to executing a **Memory Reference** instruction (i.e., an instruction that reads data from memory, writes data to memory or operates on data stored in memory).

#### Arithmetic/Logic Unit (ALU):

All processors contain an arithmetic/logic unit, which is often referred to simply as the **ALU**. The ALU, as its name implies, is that portion of the CPU hardware which

performs the arithmetic and logical operations on the binary data.

The ALU must contain an **Adder** which is capable of combining the contents of two registers in accordance with the logic of binary arithmetic. This provision permits the processor to perform arithmetic manipulations on the data it obtains from memory and from its other inputs.

Using only the basic adder a capable programmer can write routines which will subtract, multiply and divide, giving the machine complete arithmetic capabilities. In practice, however, most ALUs provide other built-in functions, including hardware subtraction, boolean logic operations, and shift capabilities.

The ALU contains **Flag Bits** which specify certain conditions that arise in the course of arithmetic and logical manipulations. Flags typically include **Carry**, **Zero**, **Sign**, and **Parity**. It is possible to program jumps which are conditionally dependent on the status of one or more flags. Thus, for example, the program may be designed to jump to a special routine if the carry bit is set following an addition instruction.

#### Control Circuitry:

The control circuitry is the primary functional unit within a CPU. Using clock inputs, the control circuitry maintains the proper sequence of events required for any processing task. After an instruction is fetched and decoded, the control circuitry issues the appropriate signals (to units both internal and external to the CPU) for initiating the proper processing action. Often the control circuitry will be capable of responding to external signals, such as an interrupt or wait request. An **Interrupt** request will cause the control circuitry to temporarily interrupt main program execution, jump to a special routine to service the interrupting device, then automatically return to the main program. A **Wait** request is often issued by a memory or I/O element that operates slower than the CPU. The control circuitry will idle the CPU until the memory or I/O port is ready with the data.

### COMPUTER OPERATIONS

There are certain operations that are basic to almost any computer. A sound understanding of these basic operations is a necessary prerequisite to examining the specific operations of a particular computer.

#### Timing:

The activities of the central processor are cyclical. The processor fetches an instruction, performs the operations required, fetches the next instruction, and so on. This orderly sequence of events requires precise timing, and the CPU therefore requires a free running oscillator clock which furnishes the reference for all processor actions. The combined fetch and execution of a single instruction is referred to as an **Instruction Cycle**. The portion of a cycle identified



clock period. As a general rule, one or more clock periods are necessary for the completion of a state, and there are several states in a cycle.

### Instruction Fetch:

The first state(s) of any instruction cycle will be dedicated to fetching the next instruction. The CPU issues a read signal and the contents of the program counter are sent to memory, which responds by returning the next instruction word. The first byte of the instruction is placed in the instruction register. If the instruction consists of more than one byte, additional states are required to fetch each byte of the instruction. When the entire instruction is present in the CPU, the program counter is incremented (in preparation for the next instruction fetch) and the instruction is decoded. The operation specified in the instruction will be executed in the remaining states of the instruction cycle. The instruction may call for a memory read or write, an input or output and/or an internal CPU operation, such as a register-to-register transfer or an add-registers operation.

### Memory Read:

An instruction **fetch** is merely a special memory read operation that brings the instruction to the CPU's instruction register. The instruction fetched may then call for data to be read from memory into the CPU. The CPU again issues a read signal and sends the proper memory address; memory responds by returning the requested word. The data received is placed in the accumulator or one of the other general purpose registers (not the instruction register).

### Memory Write:

A memory write operation is similar to a read except for the direction of data flow. The CPU issues a write signal, sends the proper memory address, then sends the data word to be written into the addressed memory location.

### Wait (memory synchronization):

As previously stated, the activities of the processor are timed by a master clock oscillator. The clock period determines the timing of all processing activity.

The speed of the processing cycle, however, is limited by the memory's **Access Time**. Once the processor has sent a read address to memory, it cannot proceed until the memory has had time to respond. Most memories are capable of responding much faster than the processing cycle requires. A few, however, cannot supply the addressed byte within the minimum time established by the processor's clock.

Therefore a processor should contain a synchronization provision, which permits the memory to request a **Wait state**. When the memory receives a read or write enable signal, it places a request signal on the processor's **READY** line, causing the CPU to idle temporarily. After the memory has

### Input/Output:

Input and Output operations are similar to memory read and write operations with the exception that a peripheral I/O device is addressed instead of a memory location. The CPU issues the appropriate input or output control signal, sends the proper device address and either receives the data being input or sends the data to be output.

Data can be input/output in either parallel or serial form. All data within a digital computer is represented in binary coded form. A binary data word consists of a group of bits; each bit is either a one or a zero. **Parallel** I/O consists of transferring all bits in the word at the same time, one bit per line. **Serial** I/O consists of transferring one bit at a time on a single line. Naturally serial I/O is much slower, but it requires considerably less hardware than does parallel I/O.

### Interrupts:

**Interrupt** provisions are included on many central processors, as a means of improving the processor's efficiency. Consider the case of a computer that is processing a large volume of data, portions of which are to be output to a printer. The CPU can output a byte of data within a single machine cycle but it may take the printer the equivalent of many machine cycles to actually print the character specified by the data byte. The CPU could then remain idle waiting until the printer can accept the next data byte. If an interrupt capability is implemented on the computer, the CPU can output a data byte then return to data processing. When the printer is ready to accept the next data byte, it can request an interrupt. When the CPU acknowledges the interrupt, it suspends main program execution and automatically branches to a routine that will output the next data byte. After the byte is output, the CPU continues with main program execution. Note that this is, in principle, quite similar to a subroutine call, except that the jump is initiated externally rather than by the program.

More complex interrupt structures are possible, in which several interrupting devices share the same processor but have different priority levels. Interruptive processing is an important feature that enables maximum utilization of a processor's capacity for high system throughput.

### Hold:

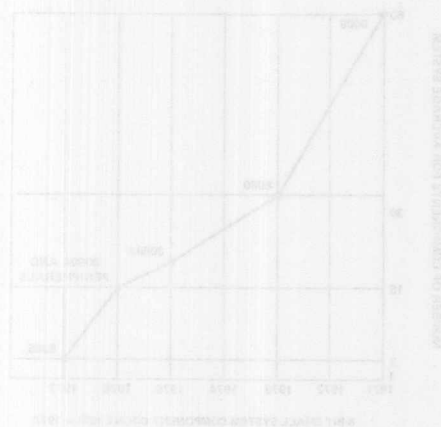
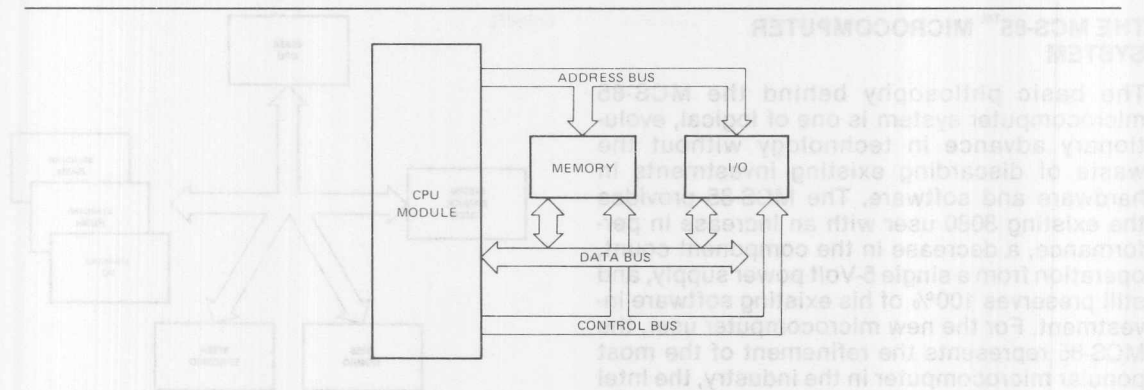
Another important feature that improves the throughput of a processor is the **Hold**. The hold provision enables **Direct Memory Access (DMA)** operations.

In ordinary input and output operations, the processor itself supervises the entire data transfer. Information to be placed in memory is transferred from the input device to the processor, and then from the processor to the designated memory location. In similar fashion, information that goes

from memory to output devices goes by way of the processor.

Some peripheral devices, however, are capable of transferring information to and from memory much faster than the processor itself can accomplish the transfer. If any appreciable quantity of data must be transferred to or from such a device, then **system throughput** will be increased by

having the device accomplish the transfer directly. The processor must temporarily suspend its operation during such a transfer, to prevent conflicts that would arise if processor and peripheral device attempted to access memory simultaneously. It is for this reason that a **hold** provision is included on some processors.



This section of the MCS-80/85 User's Manual will briefly detail the basic differences between the MCS-85 and MCS-80 families. It will illustrate both the hardware and software compatibility and also reveal some of the engineering trade-offs that were met during the design of the MCS-85. More detailed discussion of the MCS-85 bus operation and component specifications are available in Chapter 2, 3, 4, and 5. The information provided in Chapter 1 will be helpful in understanding the basic concepts and philosophies behind the MCS-85.

## EVOLUTION

In December 1977, Intel introduced the first general purpose, 8-bit microprocessor, the 8008. It was implemented in P-channel MOS technology and was packaged in a single 18 pin dual in-line package (DIP). The 8008 used standard semiconductor ROM and RAM and, for the most part, TTL components for I/O and general interface. It immediately found applications in byte-oriented and products such as terminals and computer peripherals where its instruction execution (20 micro-seconds) general

## PART 2: INTRODUCTION TO MCS-85™

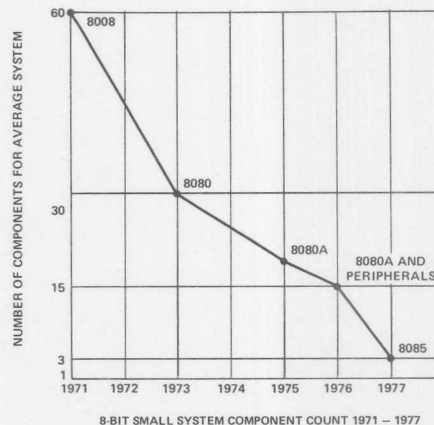
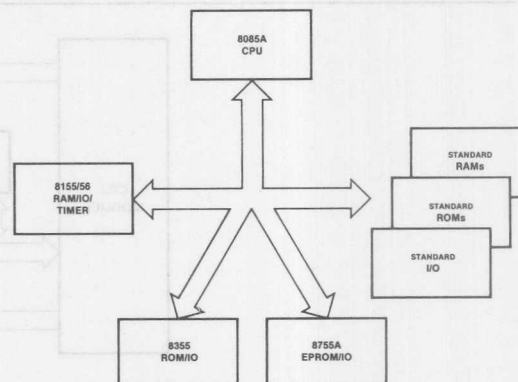
### THE MCS-85™ MICROCOMPUTER SYSTEM

The basic philosophy behind the MCS-85 microcomputer system is one of logical, evolutionary advance in technology without the waste of discarding existing investments in hardware and software. The MCS-85 provides the existing 8080 user with an increase in performance, a decrease in the component count, operation from a single 5-Volt power supply, and still preserves 100% of his existing software investment. For the new microcomputer user, the MCS-85 represents the refinement of the most popular microcomputer in the industry, the Intel 8080, along with a wealth of supporting software, documentation and peripheral components to speed the cycle from prototype to production. The same development tools that Intel has produced to support the 8080 microcomputer system can be used for the MCS-85, and additional add-on features are available to optimize system development for MCS-85.

This section of the MCS-80/85 User's Manual will briefly detail the basic differences between the MCS-85 and MCS-80 families. It will illustrate both the hardware and software compatibilities and also reveal some of the engineering trade-offs that were met during the design of the MCS-85. More detailed discussion of the MCS-85 bus operation and component specifications are available in Chapters: 2, 3, 4, and 5. The information provided in Chapter 1 will be helpful in understanding the basic concepts and philosophies behind the MCS-85.

### EVOLUTION

In December 1971, Intel introduced the first general purpose, 8-bit microprocessor, the 8008. It was implemented in P-channel MOS technology and was packaged in a single 18 pin, dual in-line package (DIP). The 8008 used standard semiconductor ROM and RAM and, for the most part, TTL components for I/O and general interface. It immediately found applications in byte-oriented end products such as terminals and computer peripherals where its instruction execution (20 micro-seconds), general



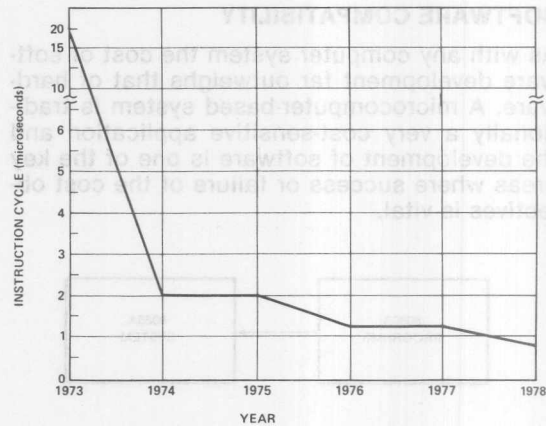
purpose organization and instruction set matched the requirements of these products. Recognizing that hardware was but a small part in the overall system picture, Intel developed both hardware and software tools for the design engineer so that the transition from prototype to production would be as simple and fast as possible. The commitment of providing a total systems approach with the 8008 microcomputer system was actually the basis for the sophisticated, comprehensive development tools that Intel has available today.

## THE 8080A MICROPROCESSOR

With the advent of high-production N-channel RAM memories and 40 pin DIP packaging, Intel designed the 8080A microprocessor. It was designed to be software compatible with the 8008 so that the existing users of the 8008 could preserve their investment in software and at the same time provide dramatically increased performance (2 micro-second instruction execution), while reducing the amount of components necessary to implement a system. Additions were made to the basic instruction set to take advantage of this increased performance and large system-type features were included on-chip such as DMA, 16-bit addressing and external stack memory so that the total spectrum of application could be significantly increased. The 8080 was first sampled in December 1973. Since that time it has become the standard of the industry and is accepted as the primary building block for more microcomputer based applications than all other microcomputer systems combined.

## A TOTAL SYSTEMS COMMITMENT

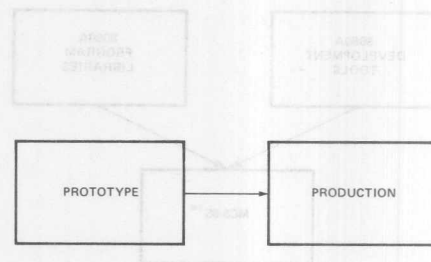
The Intel® 8080A Microcomputer System encompasses a total systems commitment to the user to fully support his needs both in developing prototype systems and reliable, high volume production. From complex MOS/LSI peripheral components to resident high level systems language (PL/M) the Intel® 8080 Microcomputer System provides the most comprehensive, effective solution to today's system problems.



The 8085A CPU is 100% software compatible with the Intel® 8080A CPU. The compatibility is at the object or "machine code" level so that existing programs written for 8080A execution will run on the 8085A as is. The value of this becomes even more evident to the user who has mask programmed ROMs and wishes to update his system without the need for new masks.

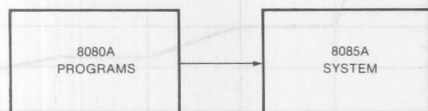
## PROGRAMMER TRAINING

A cost which is often forgotten is that of programmer training. A new or modified instruction set would require programmers to learn another set of mnemonics and greatly affect the productivity during development. The 100% compatibility of the 8085A CPU assures that no re-training effort will be required.





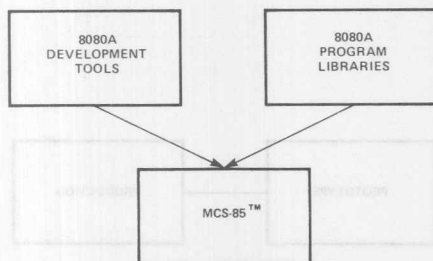
For many computer systems the cost of software development far outweighs that of hardware. A microcomputer-based system is traditionally a very cost-sensitive application and the development of software is one of the key areas where success or failure of the cost objectives is vital.



The 8085A CPU is 100% software compatible with the Intel® 8080A CPU. The compatibility is at the object or "machine code" level so that existing programs written for 8080A execution will run on the 8085A as is. The value of this becomes even more evident to the user who has mask programmed ROMs and wishes to update his system without the need for new masks.

### PROGRAMMER TRAINING

A cost which is often forgotten is that of programmer training. A new, or modified instruction set, would require programmers to relearn another set of mnemonics and greatly affect the productivity during development. The 100% compatibility of the 8085A CPU assures that no re-training effort will be required.



means that all of the software development tools that are available for the 8080A and all software libraries for 8080A will operate with the new design and thus save immeasurable cost in development and debug.

The 8085A CPU does however add two instructions to initialize and maintain hardware features of the 8085A. Two of the unused op-codes of the 8080A instruction set were designated for the addition so that 100% compatibility could be maintained.

### HARDWARE COMPATIBILITY

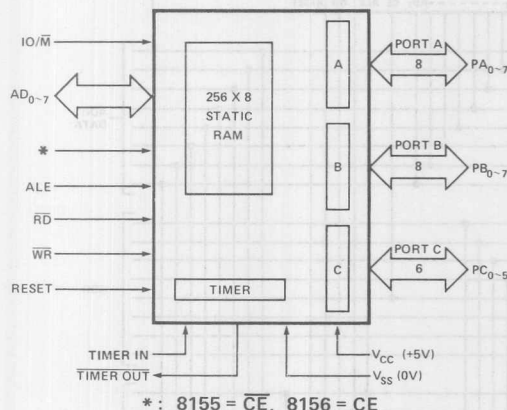
The integration of auxiliary 8080A functions, such as clock generation, system control and interrupt prioritization, dramatically reduces the amount of components necessary for most systems. In addition, the MCS-85 operates off a single +5 Volt power supply to further simplify hardware development and debug. A close examination of the AC/DC specifications of the MCS-85 systems components shows that each is specified to supply a minimum of 400 $\mu$ A of source current and a full TTL load of sink current so that a very substantial system can be constructed without the need for extra TTL buffers or drivers. Input and output voltage levels are also specified so that a minimum of 350mV noise margin is provided for reliable, high-performance operation.

### PC BOARD CONSIDERATIONS

The 8085A CPU and the 8080A are not pin-compatible due to the reduction in power supplies and the addition of integrated auxiliary features. However, the pinouts of the MCS-85 system components were carefully assigned to minimize PC board area and thus yield a smooth, efficient layout. For new designs this incompatibility of pinouts presents no problems and for upgrades of existing designs the reduction of components and board area will far offset the incompatibility.

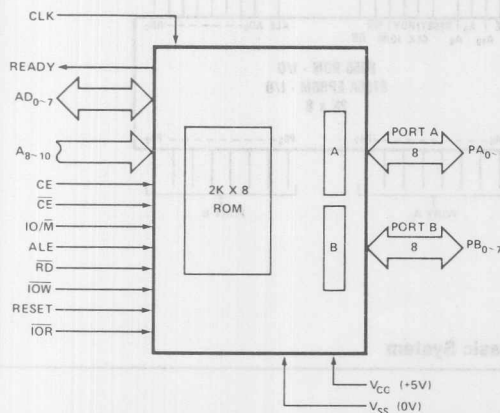
## MCS-85™ SPECIAL PERIPHERAL COMPONENTS

The MCS-85 was designed to minimize the amount of components required for most systems. Intel designed several new peripheral components that combine memory, I/O and timer functions to fulfill this requirement. These new peripheral devices directly interface to the multiplexed MCS-85 bus structure and provide new levels in system integration for today's designer.



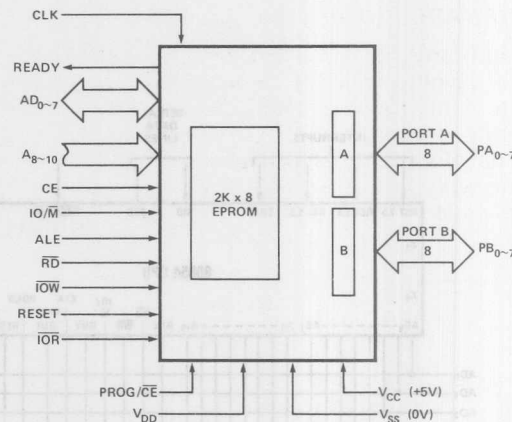
8155/8156 RAM, I/O and Timer

256 bytes RAM  
Two 8-bit ports  
One 6-bit port (programmable)  
One 14-bit programmable interval timer  
Single +5 Volt supply operation  
40 pin DIP plastic or cerdip package



8355 ROM and I/O

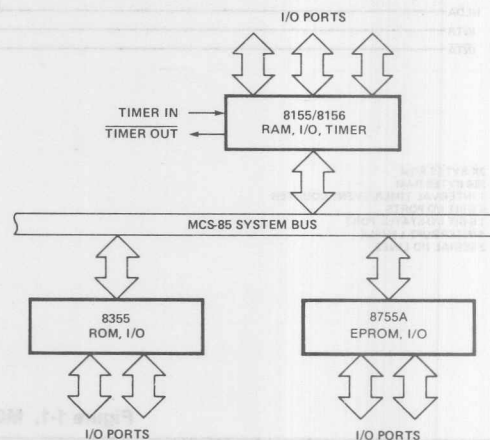
2K bytes ROM  
Two 8-bit ports (direction programmable)  
Single +5 Volt supply operation  
40 pin DIP plastic or cerdip package



8755A EPROM and I/O

Socket compatible with 8355  
2K bytes EPROM  
Two 8-bit ports (direction programmable)  
Single +5 Volt supply read operation  
U.V. Erasable  
40 pin DIP package

One of the most important advances made with the MCS-85 is the socket-compatibility of the 8355 and 8755A components. This allows the systems designer to develop and debug in erasable PROM and then, when satisfied, switch over to mask-programmed ROM 8355 with no performance degradation or board layout. It also allows quick prototype production for market impact without going to a compromise solution.



## SYSTEM EXPANSION

Each of these peripheral components has features that allow a small to medium system to be constructed without the addition of buffers and decoders to maintain the lowest possible component count.

# INTRODUCTION TO MCS-85™

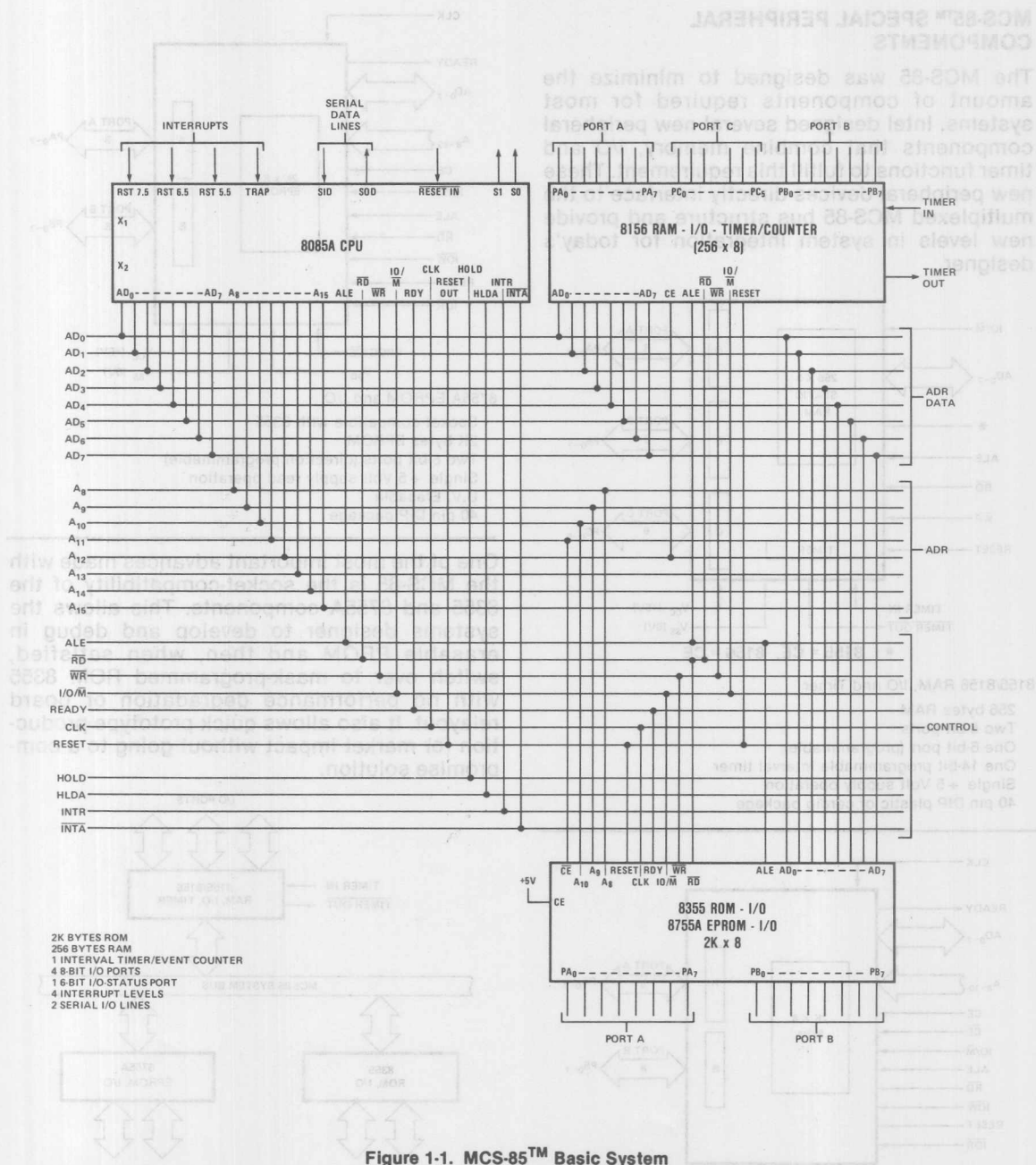


Figure 1-1. MCS-85™ Basic System

## INTERFACING TO MCS-80/85™ PROGRAMMABLE PERIPHERAL COMPONENTS

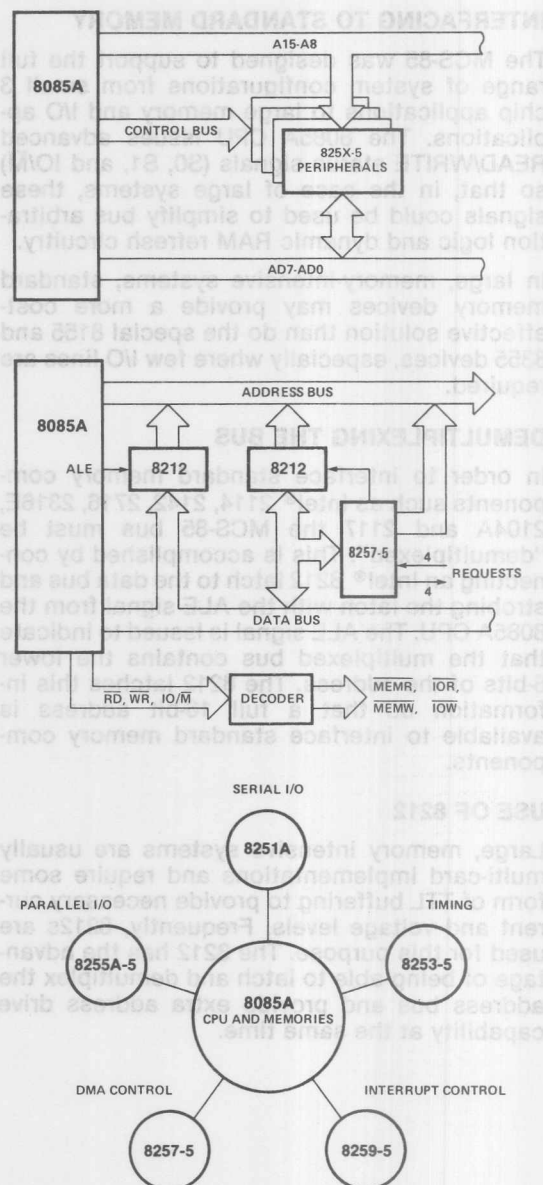
The MCS-85 shares with the MCS-80 a wide range of peripheral components that solve system problems and provide the designer with a great deal of flexibility in his I/O, Interrupt and DMA structures. The MCS-85 is directly compatible with these peripherals, and, with the exception of the 8257-5 DMA controller, needs no additional circuitry for their interface in a minimum system. The 8257-5 DMA controller uses an 8212 latch and some gating to support the multiplexed bus of MCS-85.

## PROGRAMMABLE PERIPHERALS

The list of programmable peripherals for use with the 8085A includes:

8251A	Programmable Communications Interface
8253-5	Programmable Interval Timer
8255A-5	Programmable Peripheral Interface
8257-5	Programmable DMA Controller
8259-5	Programmable Interrupt Controller
8271	Diskette Controller
8273	Synchronous Data Link Controller
8275	CRT Controller
8278	Keyboard/Display Controller
8279	Keyboard/Display Controller

The MCS-80/85 peripheral compatibility assures the designer that all new peripheral components from Intel will interface to the MCS-85 bus structure to further expand the application spectrum of MCS-85.





## INTERFACING TO STANDARD MEMORY

The MCS-85 was designed to support the full range of system configurations from small 3 chip applications to large memory and I/O applications. The 8085A CPU issues advanced READ/WRITE status signals (S0, S1, and IO/M) so that, in the case of large systems, these signals could be used to simplify bus arbitration logic and dynamic RAM refresh circuitry.

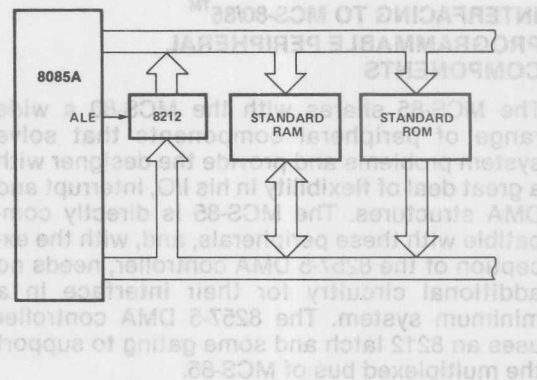
In large, memory-intensive systems, standard memory devices may provide a more cost-effective solution than do the special 8155 and 8355 devices, especially where few I/O lines are required.

## DEMULTIPLEXING THE BUS

In order to interface standard memory components such as Intel® 2114, 2142, 2716, 2316E, 2104A and 2117 the MCS-85 bus must be "demultiplexed". This is accomplished by connecting an Intel® 8212 latch to the data bus and strobing the latch with the ALE signal from the 8085A CPU. The ALE signal is issued to indicate that the multiplexed bus contains the lower 8-bits of the address. The 8212 latches this information so that a full 16-bit address is available to interface standard memory components.

## USE OF 8212

Large, memory intensive systems are usually multi-card implementations and require some form of TTL buffering to provide necessary current and voltage levels. Frequently, 8212s are used for this purpose. The 8212 has the advantage of being able to latch and demultiplex the address bus and provide extra address drive capability at the same time.



PROGRAMMABLE PERIPHERALS

The list of programmable peripherals for use with the 8085A includes:

8251A	Programmable Communications Interface
8253-5	Programmable Interval Timer
8255A-5	Programmable Peripheral Interface
8257-5	Programmable DMA Controller
8258-5	Programmable Interrupt Controller
8271	Diskette Controller
8273	Synchronous Data Link Controller
8275	CRT Controller
8278	Keyboard/Display Controller
8279	Keyboard/Display Controller

The MCS-80/85 peripheral compatibility assures the designer that all new peripheral components from Intel will interface to the MCS-85 bus structure to further expand the application spectrum of MCS-85.

## SYSTEM PERFORMANCE

The true benchmark of any microcomputer-based system is the amount of tasks that can be performed by the system in a given period of time. Increasing speed of CPU instruction execution has been the common approach to increasing system throughput but this puts a greater strain on the memory access requirement and bus operation than is usually practical for most applications. A much more desirable method would be to distribute the task-load to peripheral devices.

## DISTRIBUTED PROCESSING

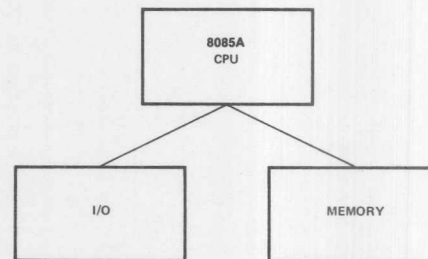
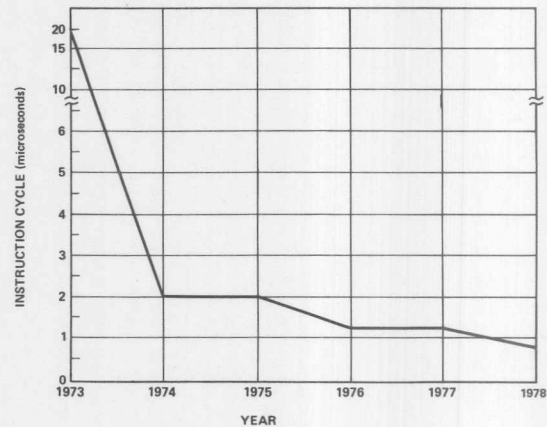
The concept of distributed task processing is not new to the computer designer, but until recently little if any task distribution was available to the microcomputer user. The use of the new programmable MCS-80/85 peripherals can relieve the central processor of many of the bookkeeping I/O and timing tasks that would otherwise have to be handled by system software.

## INSTRUCTION CYCLE/ACCESS TIME

The basic instruction cycle of the 8085A is 1.3 microseconds, the same speed as the 8080A-1. A close look at the MCS-85 bus operation shows that the access requirement for this speed is only 575 nanoseconds. The MCS-80 access requirements for this speed would be under 300 nanoseconds. This illustrates the efficiency and improved timing margins of the MCS-85 bus structure. The new 8085A-2, a high-speed selected version of the 8085A with a .8 microsecond instruction cycle, provides a 60% performance improvement over the standard 8085A.

## CONCLUSIONS: THROUGHPUT/COST

When a total system throughput/cost analysis is taken, the MCS-85 system with its advanced processor will yield the most cost-effective, reliable and producible system.





# **Chapter 2**

## **8085A**

### **Functional Description**

**MOS**

**8085A**

**MOS**

**8085A**





## CHAPTER 2

### 8085A FUNCTIONAL DESCRIPTION

#### 2.1 WHAT THE 8085A IS

The 8085A is an 8-bit general-purpose microprocessor that is very cost-effective in small systems because of its extraordinarily low hardware overhead requirements. At the same time it is capable of accessing up to 64K bytes of memory and has status lines for controlling large systems.

#### 2.2 WHAT'S IN THE 8085A

In the 8085A microprocessor are contained the functions of clock generation, system bus control, and interrupt priority selection, in addition to execution of the instruction set. (See Figure 2-1.) The 8085A transfers data on an 8-bit, bi-directional 3-state bus ( $AD_{0-7}$ ) which is time-multiplexed so as to also transmit the eight lower-order address bits. An additional eight lines ( $A_{8-15}$ ) expand the MCS-85 system memory addressing capability to 16 bits, thereby allowing 64K bytes of memory to be accessed directly by the CPU. The 8085A CPU (central processing unit) generates control signals that can be used to select appropriate external devices and

functions to perform READ and WRITE operations and also to select memory or I/O ports. The 8085A can address up to 256 different I/O locations. These addresses have the same numerical values (00 through FFH) as the first 256 memory addresses; they are distinguished by means of the I/O/M output from the CPU. You may also choose to address I/O ports as memory locations (i.e., memory-map the I/O, Section 3.2).

#### 2.2.1 Registers

The 8085A, like the 8080, is provided with internal 8-bit registers and 16-bit registers. The 8085A has eight addressable 8-bit registers. Six of them can be used either as 8-bit registers or as 16-bit register pairs. Register pairs are treated as though they were single, 16-bit registers; the high-order byte of a pair is located in the first register and the low-order byte is located in the second. In addition to the register pairs, the 8085A contains two more 16-bit registers.

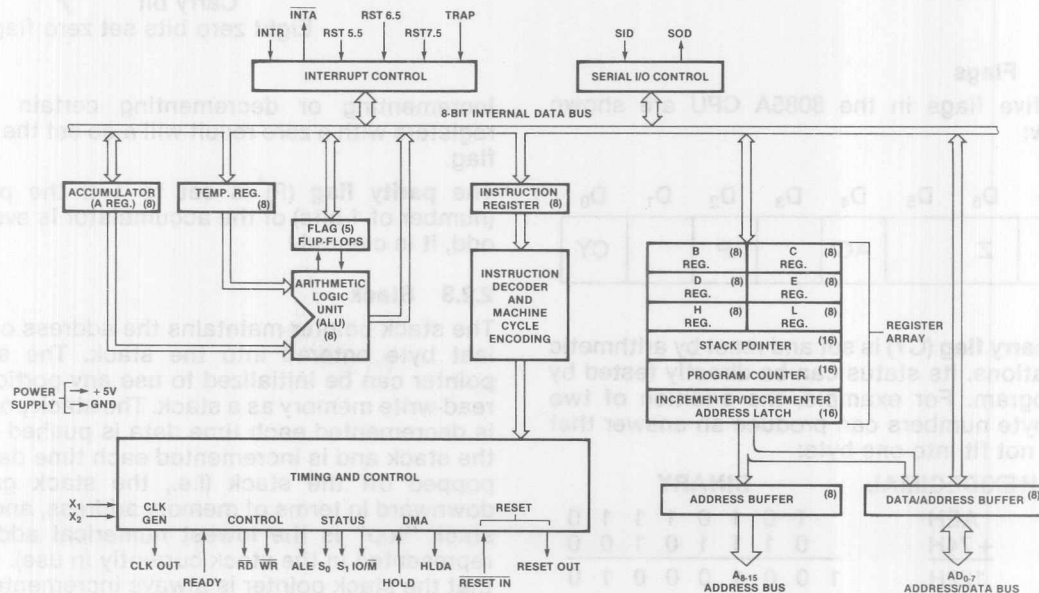


FIGURE 2-1 8085A CPU FUNCTIONAL BLOCK DIAGRAM

## FUNCTIONAL DESCRIPTION

The 8085A's CPU registers are distinguished as follows:

- The **accumulator** (ACC or A Register) is the focus of all of the accumulator instructions (Table 4-1), which include arithmetic, logic, load and store, and I/O instructions. It is an 8-bit register only. (However, see **Flags**, in this list.)
- The **program counter** (PC) always points to the memory location of the next instruction to be executed. It always contains a 16-bit address.
- **General-purpose registers** BC, DE, and HL may be used as six 8-bit registers or as three 16-bit registers, interchangeably, depending on the instruction being performed. HL functions as a **data pointer** to reference memory addresses that are either the sources or the destinations in a number of instructions. A smaller number of instructions can use BC or DE for indirect addressing.
- The **stack pointer** (SP) is a special data pointer that always points to the stack top (next available stack address). It is an indivisible 16-bit register.
- The **flag register** contains five one-bit flags, each of which records processor status information and may also control processor operation. (See following paragraph.)

### 2.2.2 Flags

The five flags in the 8085A CPU are shown below:

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
S	Z		AC		P		CY

The **carry flag** (CY) is set and reset by arithmetic operations. Its status can be directly tested by a program. For example, the addition of two one-byte numbers can produce an answer that does not fit into one byte:

HEXDECIMAL	BINARY
AEH	1 0 1 0 1 1 1 0
+ 74H	0 1 1 1 0 1 0 0
122H	1 0 0 1 0 0 0 1 0

Carry bit sets carry flag to 1

An addition operation that results in an overflow out of the high-order bit of the accumulator sets the carry flag. An addition operation that does not result in an overflow clears the carry flag. (See 8080/8085 Assembly Language Programming Manual for further details.) The carry flag also acts as a "borrow" flag for subtract operations.

The **auxiliary carry flag** (AC) indicates overflow out of bit 3 of the accumulator in the same way that the carry flag indicates overflow out of bit 7. This flag is commonly used in BCD (binary coded decimal) arithmetic.

The **sign flag** is set to the condition of the most significant bit of the accumulator following the execution of arithmetic or logic instructions. These instructions use bit 7 of data to represent the sign of the number contained in the accumulator. This permits the manipulation of numbers in the range from -128 to +127.

The **zero flag** is set if the result generated by certain instructions is zero. The zero flag is cleared if the result is not zero. A result that has a carry but has a zero answer byte in the accumulator will set both the carry flag and the zero flag. For example,

HEXDECIMAL	BINARY
A7H	1 0 1 0 0 1 1 1
+ 59H	+ 0 1 0 1 1 0 0 1
100H	1 0 0 0 0 0 0 0

Carry bit

Eight zero bits set zero flag to 1

Incrementing or decrementing certain CPU registers with a zero result will also set the zero flag.

The **parity flag** (P) is set to 1 if the parity (number of 1-bits) of the accumulator is even. If odd, it is cleared.

### 2.2.3 Stack

The stack pointer maintains the address of the last byte entered into the stack. The stack pointer can be initialized to use any portion of read-write memory as a stack. The stack pointer is decremented each time data is pushed onto the stack and is incremented each time data is popped off the stack (i.e., the stack grows downward in terms of memory address, and the stack "top" is the lowest numerical address represented in the stack currently in use). Note that the stack pointer is always incremented or decremented by two bytes since all stack operations apply to register pairs.

### 2.2.4 Arithmetic-Logic Unit (ALU)

The ALU contains the accumulator and the flag register (described in Sections 2.2.1 and 2.2.2) and some temporary registers that are inaccessible to the programmer.

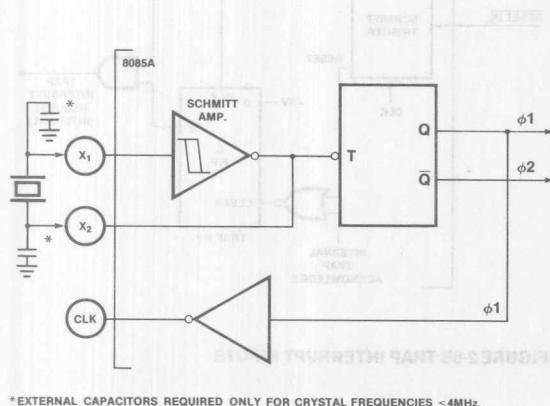
Arithmetic, logic, and rotate operations are performed by the ALU. The results of these operations can be deposited in the accumulator, or they can be transferred to the internal data bus for use elsewhere.

### 2.2.5 Instruction Register and Decoder

During an instruction fetch, the first byte of an instruction (containing the opcode) is transferred from the internal bus to the 8-bit instruction register. (See Figure 2-1.) The contents of the instruction register are, in turn, available to the instruction decoder. The output of the decoder, gated by timing signals, controls the registers, ALU, and data and address buffers. The outputs of the instruction decoder and internal clock generator generate the state and machine cycle timing signals.

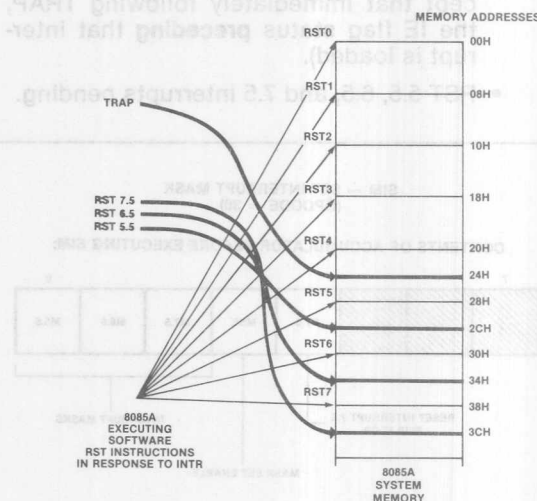
### 2.2.6 Internal Clock Generator

The 8085A CPU incorporates a complete clock generator on its chip, so it requires only the addition of a quartz crystal to establish timing for its operation. (It will accept an external clock input at its  $X_1$  input instead, however.) A suitable crystal for the standard 8085A must be parallel-resonant at a fundamental of 6.25 MHz or less, twice the desired internal clock frequency. The 8085A-2 will operate with crystal of up to 10 MHz. The functions of the 8085A internal clock generator are shown in Figure 2-2. A Schmitt trigger is used interchangeably as oscillator or



### FIGURE 2-2 8085A CLOCK LOGIC

as input conditioner, depending upon whether a crystal or an external source is used. The clock circuitry generates two nonoverlapping internal clock signals,  $\phi_1$  and  $\phi_2$  (see Figure 2-2).  $\phi_1$  and  $\phi_2$  control the internal timing of the 8085A and are not directly available on the outside of the chip. The external pin CLK is a buffered, inverted version of  $\phi_1$ . CLK is half the frequency of the crystal input signal and may be used for clocking other devices in the system.



### FIGURE 2-3 8085A HARDWARE AND SOFTWARE RST BRANCH LOCATIONS

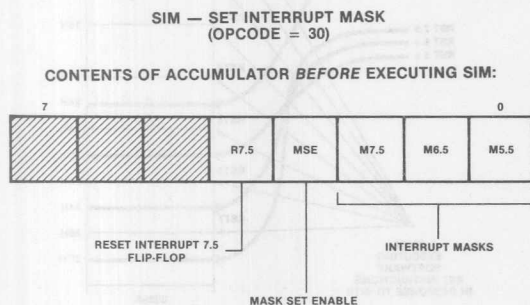
### 2.2.7 Interrupts

The five hardware interrupt inputs provided in the 8085A are of three types. INTR is identical with the 8080A INT line in function; i.e., it is maskable (can be enabled or disabled by EI or DI software instructions), and causes the CPU to fetch in an RST instruction, externally placed on the data bus, which vectors a branch to any one of eight fixed memory locations (Restart addresses). (See Figure 2-3.) INTR can also be controlled by the 8259 programmable interrupt controller, which generates CALL instructions instead of RSTs, and can thus vector operation of the CPU to a preprogrammed subroutine located anywhere in your system's memory map. The RST 5.5, RST 6.5, and RST 7.5 hardware interrupts are different in function in that they are maskable through the use of the SIM

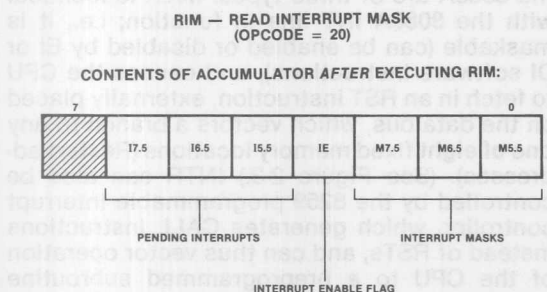


mask flags based on data in the accumulator. (See Figure 2-4.) You may read the status of the interrupt mask previously set by performing a RIM instruction. Its execution loads into the accumulator the following information. (See Figure 2-5.)

- Current interrupt mask status for the RST 5.5, 6.5, and 7.5 hardware status.
- Current interrupt enable flag status (except that immediately following TRAP, the IE flag status **preceding** that interrupt is loaded).
- RST 5.5, 6.5, and 7.5 interrupts pending.



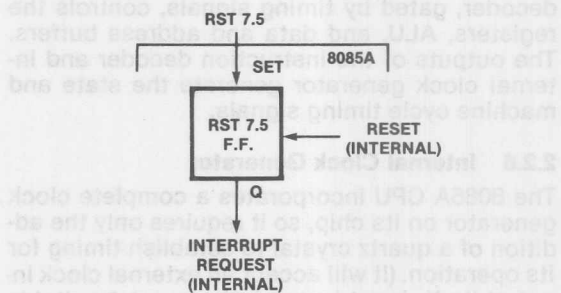
**FIGURE 2-4 INTERRUPT MASKS SET USING SIM INSTRUCTION**



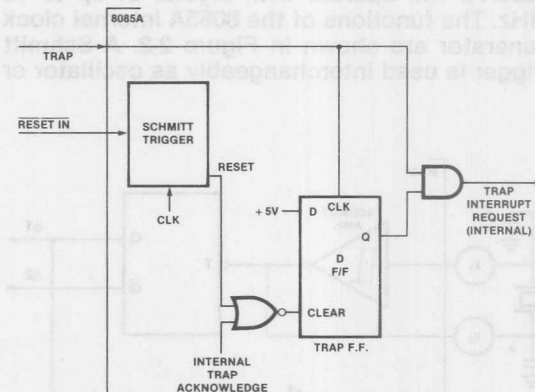
**FIGURE 2-5 RIM — READ INTERRUPT MASK**

tions, respectively. INTR, RST 5.5, and RST 6.5 are level-sensitive, meaning that these inputs may be acknowledged by the processor when they are held at a high level. RST 7.5 is edge-sensitive, meaning that an internal flip-flop in the 8085A registers the occurrence of an interrupt the instant a rising edge appears on the RST 7.5 input line. This input need not be held high; the flip-flop will remain set until it is cleared by one of three possible actions:

- The 8085A responds to the interrupt, and sends an internal reset signal to the RST 7.5 flip-flop. (See Figure 2-6A.)



**FIGURE 2-6A RST 7.5 FLIP FLOP**



**FIGURE 2-6B TRAP INTERRUPT INPUTS**

**FIGURE 2-6 RST 7.5 AND TRAP INTERRUPT INPUTS**

## FUNCTIONAL DESCRIPTION

- The 8085A, before responding to the RST 7.5 interrupt, receives a RESET IN signal from an external source; this also activates the internal reset.
- The 8085A executes a SIM instruction, with accumulator bit 4 previously set to 1. (See Figure 2-4.)

The third type of hardware interrupt is TRAP. This input is not subject to any mask or interrupt enable/disable instruction. The receipt of a positive-going edge on the TRAP input triggers the processor's hardware interrupt sequence, but the pulse must be held high until acknowledged internally (see Figure 2-6B).

The sampling of all interrupts occurs on the descending edge of CLK, one cycle before the end of the instruction in which the interrupt input is activated. To be recognized, a valid interrupt must occur at least 160 ns before sampling time in the 8085A, or 150 ns in the 8085A-2. This means that to guarantee being recognized, RST 5.5 and 6.5 and TRAP need to be held on for at least 17 clock states plus 160 ns (150 for 8085A-2), assuming that the interrupt might arrive just barely too late to be acknowledged during a particular instruction, and that the following instruction might be an 18-state CALL. This timing assumes no WAIT or HOLD cycles are used.

The way interrupt masks are set and read is described in Chapter 4 under the RIM (read in-

terrupt mask) and SIM (set interrupt mask) instruction listings. Interrupt functions and their priorities are shown in the table that follows.

Name	Priority	Address (1) Branched to when inter- rupt occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled
RST 7.5	2	3CH	Rising edge (latched)
RST 6.5	3	34H	High level until sam- pled
RST 5.5	4	2CH	High level until sam- pled
INTR	5	(2)	High level until sam- pled

### NOTES:

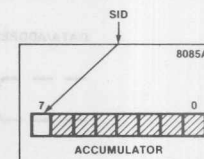
- (1) In the case of TRAP and RST 5.5-7.5, the contents of the Program Counter are pushed onto the stack before the branch occurs.
- (2) Depends on the instruction that is provided to the 8085A by the 8259 or other circuitry when the interrupt is acknowledged.

### 2.2.8 Serial Input and Output

The SID and SOD pins help to minimize chip count in small systems by providing for easy interface to a serial port using software for timing and for coding and decoding of the data. Each time a RIM instruction is executed, the status of the SID pin is read into bit 7 of the accumulator. RIM is thus a dual-purpose instruction. (See Chapter 4.) In similar fashion, SIM is used to latch bit 7 of the accumulator out to the SOD output via an internal flip-flop, providing that bit 6 of the accumulator is set to 1. (See Figure 2-7.) Section 2.3.8 describes SID and SOD timing.

SID can also be used as a general purpose TEST input and SOD can serve as a one-bit control output.

EFFECT OF RIM INSTRUCTION



EFFECT OF SIM INSTRUCTION

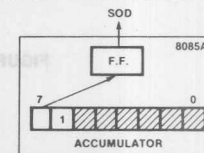


FIGURE 2-7 EFFECT OF RIM AND SIM INSTRUCTIONS ON SERIAL DATA LINES

## FUNCTIONAL DESCRIPTION

### 2.3 HOW THE MCS-85 SYSTEM WORKS

The 8085A CPU generates signals that tell peripheral devices what type of information is on the multiplexed Address/Data bus and from that point on the operation is almost identical to the MCS-80™ CPU Group. A multiplexed bus structure was chosen because it freed device pins so that more functions could be integrated on the 8085A and other components of the family. The multiplexed bus is designed to allow complete compatibility to existing peripheral

components with improved timing margins and access requirements. (See Figure 2-8.)

To enhance the system integration of MCS-85, several special components with combined memory and I/O were designed. These new devices directly interface to the multiplexed bus of the 8085A. The pin locations of the 8085A and the special peripheral components are assigned to minimize PC board area and to allow for efficient layout. The details on peripheral components are contained in subsequent paragraphs of this chapter and in Chapters 5 and 6.

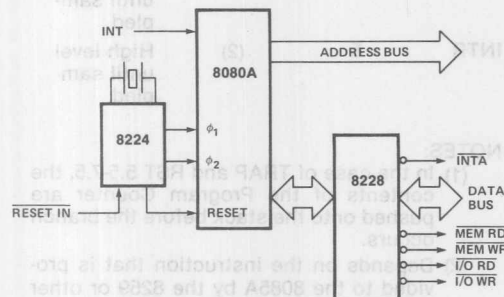


FIGURE 2-8A MCS-80™ CPU GROUP

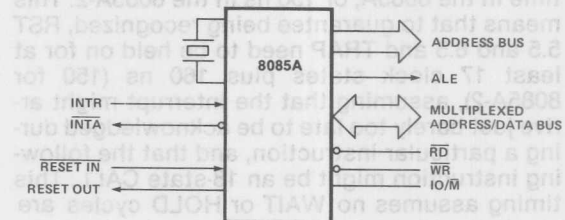


FIGURE 2-8B MCS-85™ CPU/8085A (MCS-80 COMPATIBLE FUNCTIONS)

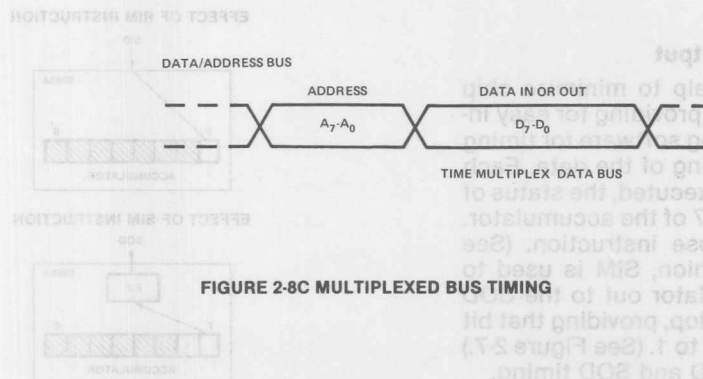


FIGURE 2-8C MULTIPLEXED BUS TIMING

FIGURE 2-8 BASIC CPU FUNCTIONS

## 2.3.1 Multiplexed Bus Cycle Timing

The execution of any 8085A program consists of a sequence of READ and WRITE operations, of which each transfers a byte of data between the 8085A and a particular memory or I/O address. These READ and WRITE operations are the only communication between the processor and the other components, and are all that is necessary to execute any instruction or program.

Each READ or WRITE operation of the 8085A is referred to as a machine cycle. The execution of each instruction by the 8085A consists of a sequence of from one to five machine cycles, and each machine cycle consists of a minimum of from three to six clock cycles (also referred to as T states). Consider the case of the Store Accumulator Direct (STA) instruction, shown in Figure 2-9. The STA instruction causes the contents of the accumulator to be stored at the direct address specified in the second and third bytes of the instruction. During the first machine cycle ( $M_1$ ), the CPU puts the contents of the program counter (PC) on the address bus and performs a MEMORY READ cycle to read from memory the opcode of the next instruction (STA). The  $M_1$  machine cycle is also referred to as the OPCODE FETCH cycle, since it fetches the operation code of the next instruction. In the fourth clock cycle ( $T_4$ ) of  $M_1$ , the CPU interprets the data read in and recognizes it as the opcode of the STA instruction. At this point the

CPU knows that it must do three more machine cycles (two MEMORY READs and one MEMORY WRITE) to complete the instruction.

The 8085A then increments the program counter so that it points to the next byte of the instruction and performs a MEMORY READ machine cycle ( $M_2$ ) at address (PC + 1). The accessed memory places the addressed data on the data bus for the CPU. The 8085A temporarily stores this data (which is the low-order byte of the direct address) internally in the CPU. The 8085A again increments the program counter to location (PC + 2) and reads from memory ( $M_3$ ) the next byte of data, which is the high-order byte of the direct address.

At this point, the 8085A has accessed all three bytes of the STA instruction, which it must now execute. The execution consists of placing the data accessed in  $M_2$  and  $M_3$  on the address bus, then placing the contents of the accumulator on the data bus, and then performing a MEMORY WRITE machine cycle ( $M_4$ ). When  $M_4$  is finished, the CPU will fetch ( $M_1$ ) the first byte of the next instruction and continue from there.

### State Transition Sequence

As the preceding example shows, the execution of an instruction consists of a series of machine cycles whose nature and sequence is determined by the opcode accessed in the  $M_1$

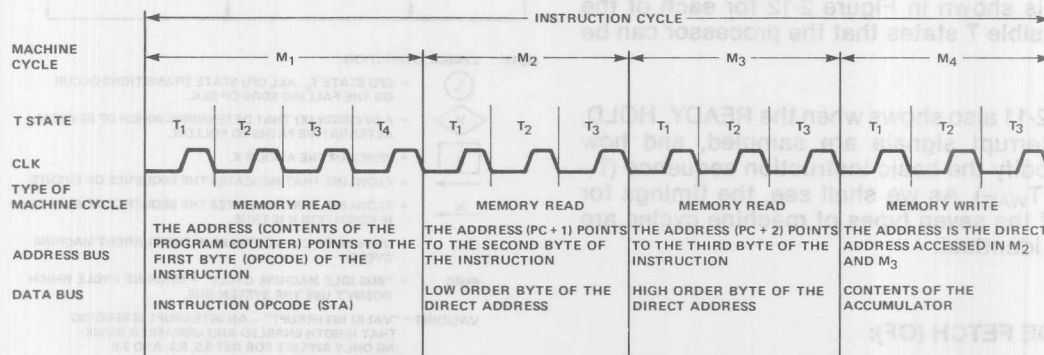


FIGURE 2-9 CPU TIMING FOR STORE ACCUMULATOR DIRECT (STA) INSTRUCTION

## FUNCTIONAL DESCRIPTION

MACHINE CYCLE		STATUS			CONTROL		
		IO/M	S1	S0	RD	WR	INTA
OPCODE FETCH (OF)		0	1	1	0	1	1
MEMORY READ (MR)		0	1	0	0	1	1
MEMORY WRITE (MW)		0	0	1	1	0	1
I/O READ (IOR)		1	1	0	0	1	1
I/O WRITE (IOW)		1	0	1	1	0	1
INTR ACKNOWLEDGE (INA)		1	1	1	1	1	0
BUS IDLE (BI): DAD		0	1	0	1	1	1
BUS IDLE (BI): INA(RST/TRAP)		1	1	1	1	1	1
HALT		TS	0	0	TS	TS	1

0 = Logic "0". 1 = Logic "1". TS = High Impedance X = Unspecified

**FIGURE 2-10 8085A MACHINE CYCLE CHART**

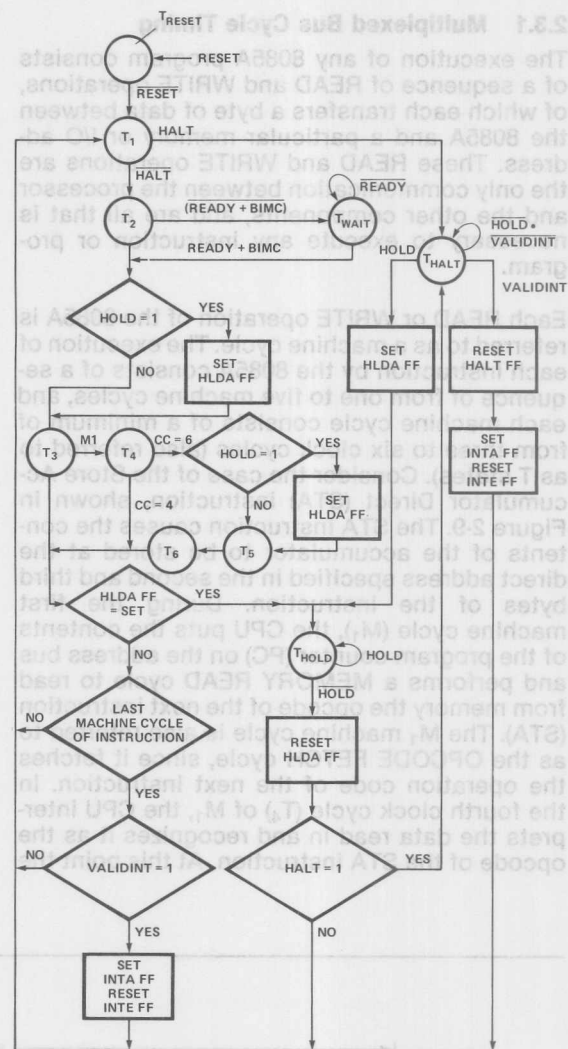
machine cycle. While no one instruction cycle will consist of more than five machine cycles, every machine cycle will be one of the seven types listed in Figure 2-10. These seven types of machine cycles can be differentiated by the state of the three status lines (IO/M, S<sub>0</sub>, and S<sub>1</sub>) and the three control signals (RD, WR, and INTA).

Most machine cycles consist of three T states, (cycles of the CLK output) with the exception of OPCODE FETCH, which normally has either four or six T states. The actual number of states required to perform any instruction depends on the instruction being executed, the particular machine cycle within the instruction cycle, and the number of WAIT and HOLD states inserted into each machine cycle through the use of the READY and HOLD inputs of the 8085A. The state transition diagram in Figure 2-11 illustrates how the 8085A proceeds in the course of a machine cycle. The state of various status and control signals, as well as the system buses, is shown in Figure 2-12 for each of the ten possible T states that the processor can be in.

Figure 2-11 also shows when the READY, HOLD, and interrupt signals are sampled, and how they modify the basic instruction sequence (T<sub>1</sub>, T<sub>6</sub> and T<sub>WAIT</sub>). As we shall see, the timings for each of the seven types of machine cycles are almost identical.

### OPCODE FETCH (OF):

The OPCODE FETCH (OF) machine cycle is unique in that it has more than three clock cycles. This is because the CPU must interpret the opcode accessed in T<sub>1</sub>, T<sub>2</sub>, and T<sub>3</sub> before it can decide what to do next.



NOTE: SYMBOL DEFINITION

- $T_n$  = CPU STATE  $T_n$ . ALL CPU STATE TRANSITIONS OCCUR ON THE FALLING EDGE OF CLK.
- X = A DECISION (X) THAT DETERMINES WHICH OF SEVERAL ALTERNATIVE PATHS TO FOLLOW.
- X = PERFORM THE ACTION X.
- X = FLOWLINE THAT INDICATES THE SEQUENCE OF EVENTS.
- X = FLOWLINE THAT INDICATES THE SEQUENCE OF EVENTS IF CONDITION X IS TRUE.
- CC = NUMBER OF CLOCK CYCLES IN THE CURRENT MACHINE CYCLE.
- BIMC = "BUS IDLE MACHINE CYCLE" = MACHINE CYCLE WHICH DOESN'T USE THE SYSTEM BUS.
- VALIDINT = "VALID INTERRUPT" - AN INTERRUPT IS PENDING THAT IS BOTH ENABLED AND UNMASKED (MASKING ONLY APPLIES FOR RST 5.5, 6.5, AND 7.5 INPUTS).
- HLDA FF = INTERNAL HOLD ACKNOWLEDGE FLIP FLOP. NOTE THAT THE 8085A SYSTEM BUSES ARE 3-STATE ONE CLOCK CYCLE AFTER THE HLDA FLIP FLOP IS SET.

**FIGURE 2-11 8085A CPU STATE TRANSITION**



## FUNCTIONAL DESCRIPTION

Machine State	Status & Buses				Control		
	S1,S0	IO/M	A <sub>8</sub> -A <sub>15</sub>	AD <sub>0</sub> -AD <sub>7</sub>	RD,WR	INTA	ALE
T <sub>1</sub>	X	X	X	X	1	1	1 <sup>†</sup>
T <sub>2</sub>	X	X	X	X	X	X	0
T <sub>WAIT</sub>	X	X	X	X	X	X	0
T <sub>3</sub>	X	X	X	X	X	X	0
T <sub>4</sub>	1	0*	X	TS	1	1	0
T <sub>5</sub>	1	0*	X	TS	1	1	0
T <sub>6</sub>	1	0*	X	TS	1	1	0
T <sub>RESET</sub>	X	TS	TS	TS	TS	1	0
T <sub>HALT</sub>	0	TS	TS	TS	TS	1	0
T <sub>HOLD</sub>	X	TS	TS	TS	TS	1	0

0 = Logic "0" 1 = Logic "1" TS = High Impedance X = Unspecified

<sup>†</sup>ALE not generated during 2nd and 3rd machine cycles of DAD instruction.

\*IO/M = 1 during T<sub>4</sub>-T<sub>6</sub> states of RST and INA cycles.

**FIGURE 2-12 8085A MACHINE STATE CHART**

Figure 2-13 shows the timing relationships for an OF machine cycle. The particular instruction illustrated is DCX, whose timing for OF differs from other instructions in that it has six T states, while some instructions require only four T states for OF. In this discussion, as well as the following discussions, only the relative timing of the signals will be discussed; for the actual timings, refer to the data sheets of the individual parts in Chapters 5 and 6.

The first thing that the 8085A does at the beginning of every machine cycle is to send out three status signals (IO/M, S1, S0) that define what type of machine cycle is about to take place. The IO/M signal identifies the machine cycle as being either a memory reference or input/output operation. The S1 status signal identifies whether the cycle is a READ or WRITE operation. The S0 and S1 status signals can be used together (see Figure 2-10) to identify READ, WRITE, or OP CODE FETCH machine cycles as well as the HALT state. Referring to Figure 2-13, the 8085A will send out IO/M = 0, S1 = 1, S0 = 1 at the beginning of the machine cycle to identify it as a READ from a memory location to obtain an opcode; in other words, it identifies the machine cycle as an OP CODE FETCH cycle.

The 8085A also sends out a 16-bit address at the beginning of every machine cycle to identify the particular memory location or I/O port that the machine cycle applies to. In the case of an OF cycle, the contents of the program counter is placed on the address bus. The high order byte (PCH) is placed on the A<sub>8</sub>-A<sub>15</sub> lines, where it will stay until at least T<sub>4</sub>. The low order byte (PCL) is placed on the AD<sub>0</sub>-AD<sub>7</sub> lines, whose three-state drivers are enabled if not found already on. Unlike the upper address lines, however, the information on the lower address lines will remain there for only one clock cycle, after which the drivers will go to their high impedance state, indicated by a dashed line in Figure 2-13. This is necessary because the AD<sub>0</sub>-AD<sub>7</sub> lines are time multiplexed between the address and data buses. During T<sub>1</sub> of every machine cycle, AD<sub>0</sub>-AD<sub>7</sub> output the lower 8-bits of address after which AD<sub>0</sub>-AD<sub>7</sub> will either output the desired data for a WRITE operation or the drivers will float (as is the case for the OF cycle), allowing the external device to drive the lines for a READ operation.

Since the address information on AD<sub>0</sub>-AD<sub>7</sub> is of a transitory nature, it must be latched either internally in special multiplexed-bus components like the 8155 or externally in parts like the 8212 8-bit latch. (See Chapter 3.) The 8085A provides a special timing signal, ADDRESS LATCH ENABLE (ALE), to facilitate the latching of A<sub>0</sub>-A<sub>7</sub>; ALE is present during T<sub>1</sub> of every machine cycle.

After the status signals and address have been sent out and the AD<sub>0</sub>-AD<sub>7</sub> drivers have been disabled, the 8085A provides a low level on RD to enable the addressed memory device. The device will then start driving the AD<sub>0</sub>-AD<sub>7</sub> lines; this is indicated by the dashed line turning into a solid line in Figure 2-13. After a period of time (which is the access time of the memory) valid data will be present on AD<sub>0</sub>-AD<sub>7</sub>. The 8085A during T<sub>3</sub> will load the memory data on AD<sub>0</sub>-AD<sub>7</sub> into its instruction register and then raise RD to the high level, disabling the addressed memory device. At this point, the 8085A will have finished accessing the opcode of the instruction. Since this is the first machine cycle (M<sub>1</sub>) of the instruction, the CPU will automatically step to T<sub>4</sub>, as shown in Figure 2-11.

During T<sub>4</sub>, the CPU will decode the opcode in the instruction register and decide whether to enter T<sub>5</sub> on the next clock or to start a new machine cycle and enter T<sub>1</sub>. In the case of the DCX instruction shown in Figure 2-13, it will enter T<sub>5</sub> and then T<sub>6</sub> before going to T<sub>1</sub>.

## FUNCTIONAL DESCRIPTION

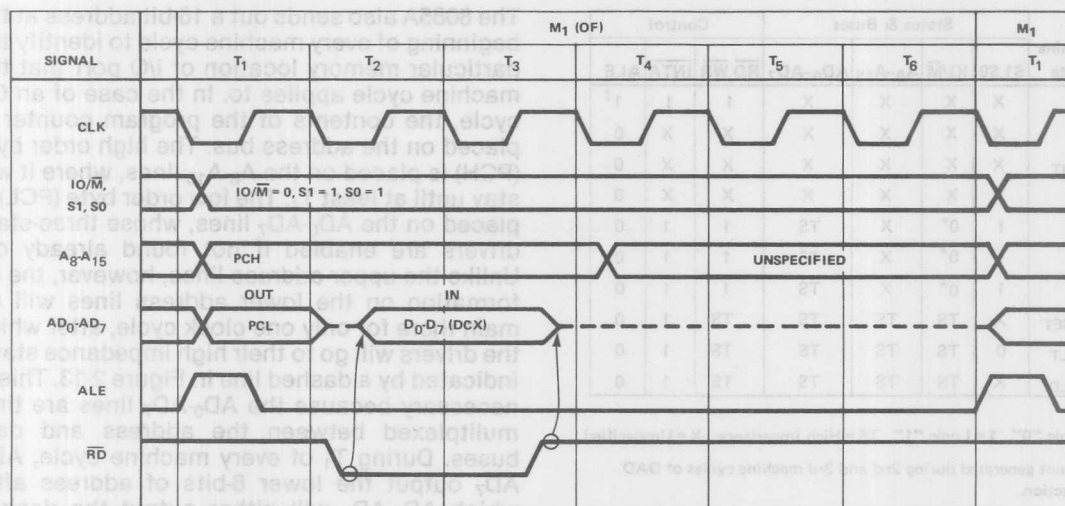


FIGURE 2-13 OP CODE FETCH MACHINE CYCLE (OF DCX INSTRUCTION)

During T<sub>5</sub> and T<sub>6</sub>, of DCX, the CPU will decrement the designated register. Since the A<sub>8</sub>-A<sub>15</sub> lines are driven by the address latch circuits, which are part of the incrementer/decrementer logic, the A<sub>8</sub>-A<sub>15</sub> lines may change during T<sub>5</sub> and T<sub>6</sub>. Because the value of A<sub>8</sub>-A<sub>15</sub> can vary during T<sub>4</sub>-T<sub>6</sub>, it is most important that all memory and I/O devices on the system bus qualify their selection with  $\overline{RD}$ . If they don't use  $\overline{RD}$ , they may be spuriously selected. Moreover, with a linear selection technique (Chapter 3), two or more devices could be simultaneously enabled, which could be potentially damaging. The generation of spurious addresses can also occur momentarily at address bus transitional periods in T<sub>1</sub>. Therefore, the selection of all memory and I/O devices must be qualified with  $\overline{RD}$  or  $\overline{WR}$ . Many new memory devices like the 8155 and 8355 have the  $\overline{RD}$  input that internally is used to enable the data bus outputs, removing the need for externally qualifying the chip enable input with  $\overline{RD}$ .

Figure 2-14 is identical to Figure 2-13 with one exception, which is the use of the READY line. As we can see in Figure 2-11, when the CPU is in T<sub>2</sub>, it examines the state of the READY line. If the READY line is high, the CPU will proceed to T<sub>3</sub> and finish executing the instruction. If the READY line is low, however, the CPU will enter T<sub>WAIT</sub> and stay there indefinitely until READY goes high. When the READY line does go high, the CPU will exit T<sub>WAIT</sub> and enter T<sub>3</sub>, in order to complete the machine cycle. As shown in

Figure 2-14, the external effect of using the READY line is to preserve the exact state of the processor signals at the end of T<sub>2</sub> for an integral number of clock periods, before finishing the machine cycle. This "stretching" of the system timing has the further effect of increasing the allowable access time for memory or I/O devices. By inserting T<sub>WAIT</sub> states, the 8085A can accommodate even the slowest of memories. Another common use of the READY line is to single-step the processor with a manual switch.

### 2.3.2 Read Cycle Timing MEMORY READ (MR):

Figure 2-15 shows the timing of two successive MEMORY READ (MR) machine cycles, the first without a T<sub>WAIT</sub> state and the second with one T<sub>WAIT</sub> state. The timing during T<sub>1</sub>-T<sub>3</sub> is absolutely identical to the OP CODE FETCH machine cycle, with the exception that the status sent out during T<sub>1</sub> is IO/M = 0, S1 = 1, S0 = 0, identifying the cycles as a READ from a memory location. This differs from Figure 2-13 only in that S0 = 1 for an OF cycle, identifying that cycle as an OP CODE FETCH operation. Otherwise, the two cycles are identical during T<sub>1</sub>-T<sub>3</sub>.

A second difference occurs at the end of T<sub>3</sub>. As shown in Figure 2-11, the CPU always goes to T<sub>4</sub> from T<sub>3</sub> during M<sub>1</sub>, which is always an OF cycle. During all other machine cycles, the CPU will always go from T<sub>3</sub> to T<sub>1</sub> of the next machine cycle.

## FUNCTIONAL DESCRIPTION

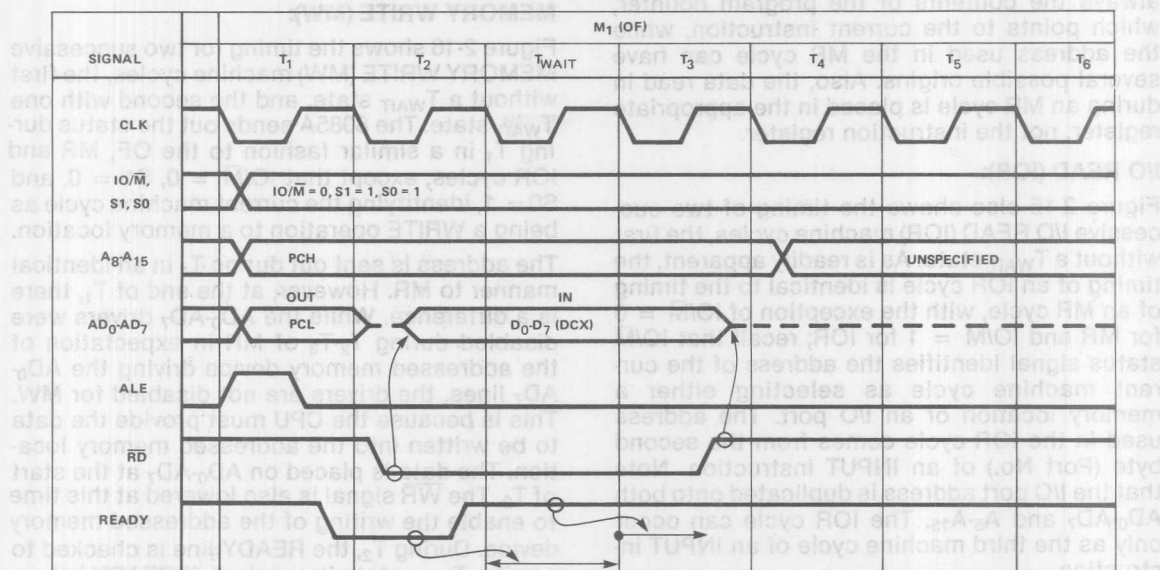


FIGURE 2-14 OPCODE FETCH MACHINE CYCLE WITH ONE WAIT STATE

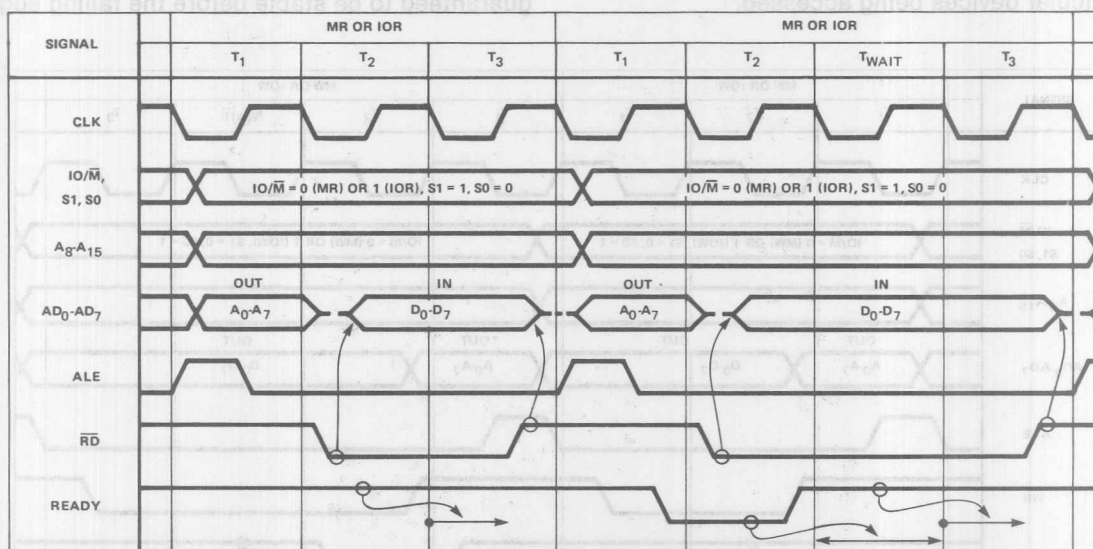


FIGURE 2-15 MEMORY READ (OR I/O READ) MACHINE CYCLES (WITH AND WITHOUT WAIT STATES)

which points to the current instruction, while the address used in the MR cycle can have several possible origins. Also, the data read in during an MR cycle is placed in the appropriate register, not the instruction register.

#### I/O READ (IOR):

Figure 2-15 also shows the timing of two successive I/O READ (IOR) machine cycles, the first without a  $T_{WAIT}$  state. As is readily apparent, the timing of an IOR cycle is identical to the timing of an MR cycle, with the exception of  $IO/\overline{M} = 0$  for MR and  $IO/\overline{M} = 1$  for IOR; recall that  $IO/\overline{M}$  status signal identifies the address of the current machine cycle as selecting either a memory location or an I/O port. The address used in the IOR cycle comes from the second byte (Port No.) of an INPUT instruction. Note that the I/O port address is duplicated onto both  $AD_0-AD_7$  and  $A_8-A_{15}$ . The IOR cycle can occur only as the third machine cycle of an INPUT instruction.

Note that the READY signal can be used to generate  $T_{WAIT}$  states for I/O devices as well as memory devices. By gating the READY signal with the proper status lines, one could generate  $T_{WAIT}$  states for memory devices only or for I/O devices only. By gating in the address lines, one can further qualify  $T_{WAIT}$  state generation by the particular devices being accessed.

Figure 2-16 shows the timing for two successive MEMORY WRITE (MW) machine cycles, the first without a  $T_{WAIT}$  state, and the second with one  $T_{WAIT}$  state. The 8085A sends out the status during  $T_1$  in a similar fashion to the OF, MR and IOR cycles, except that  $IO/\overline{M} = 0$ ,  $S_1 = 0$ , and  $S_0 = 1$ , identifying the current machine cycle as being a WRITE operation to a memory location.

The address is sent out during  $T_1$  in an identical manner to MR. However, at the end of  $T_1$ , there is a difference. While the  $AD_0-AD_7$  drivers were disabled during  $T_2-T_3$  of MR in expectation of the addressed memory device driving the  $AD_0-AD_7$  lines, the drivers are not disabled for MW. This is because the CPU must provide the data to be written into the addressed memory location. The data is placed on  $AD_0-AD_7$  at the start of  $T_2$ . The  $\overline{WR}$  signal is also lowered at this time to enable the writing of the addressed memory device. During  $T_2$ , the READY line is checked to see if a  $T_{WAIT}$  state is required. If READY is low,  $T_{WAIT}$  states are inserted until READY goes high. During  $T_3$ , the  $\overline{WR}$  line is raised, disabling the addressed memory device and thereby terminating the WRITE operation. The contents of the address and data lines are not changed until the next  $T_1$ , which directly follows.

Note that the data on  $AD_0-AD_7$  is not guaranteed to be stable before the falling edge

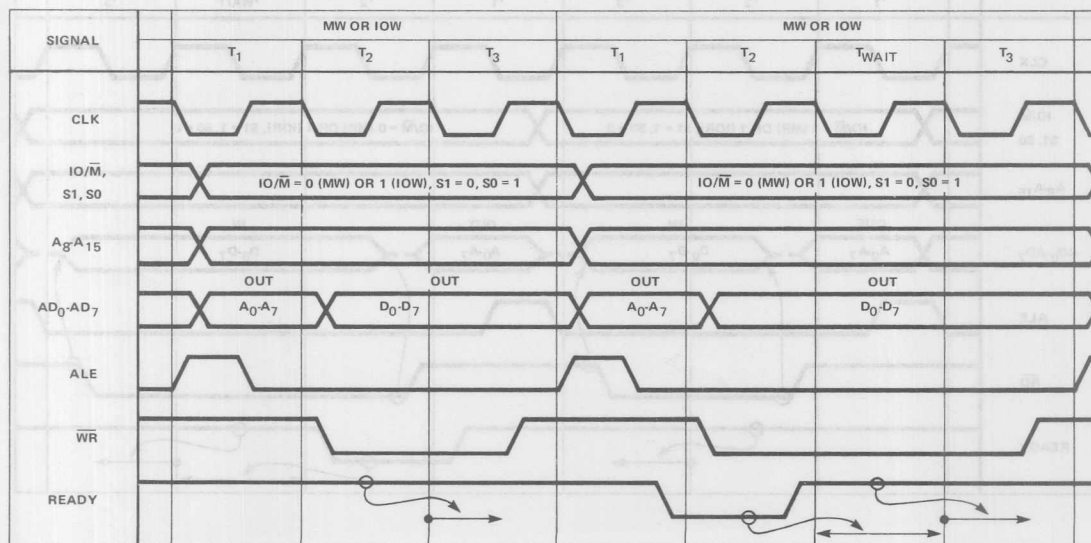


FIGURE 2-16 MEMORY WRITE (OR I/O WRITE) MACHINE CYCLES (WITH AND WITHOUT WAIT STATES)



## FUNCTIONAL DESCRIPTION

of  $\overline{WR}$ . The  $AD_0-AD_7$  lines are guaranteed to be stable both before and after the rising edge of  $\overline{WR}$ .

### I/O WRITE (IOW):

As Figure 2-16 shows, the timing for an I/O WRITE (IOW) machine cycle is the same as an MW machine cycle except that  $IO/\overline{M} = 0$  during the MW cycle and  $IO/\overline{M} = 1$  during the IOW cycle.

As with the IOR cycle discussed previously, the address used in an IOW cycle is the I/O port number which is duplicated on both the high and low bytes of the address bus. In the case of IOW, the port number comes from the second byte of an OUTPUT instruction as the instruction is executed.

### 2.3.4 Interrupt Acknowledge (INA) Timing

Figures 2-17 and 2-18 (a continuation of 2-17) depict the course of action the CPU takes in response to a high level on the INTR line if the INTE FF (interrupt enable flip-flop) has been set

by the EI instruction. The status of the TRAP and RST pins as well as INTR is sampled during the second clock cycle before  $M_1 \cdot T_1$ . If INTR was the only valid interrupt and if INTE FF is set, then the CPU will reset INTE FF and then enter an INTERRUPT ACKNOWLEDGE (INA) machine cycle. The INA cycle is identical to an OF cycle with two exceptions.  $\overline{INTA}$  is sent out instead of  $\overline{RD}$ . Also,  $IO/\overline{M} = 1$  during INA, whereas  $IO/\overline{M} = 0$  for OF. Although the contents of the program counter are sent out on the address lines, the address lines can be ignored.

When  $\overline{INTA}$  is sent out, the external interrupt logic must provide the opcode of an instruction to execute. The opcode is placed on the data bus and read in by the processor. If the opcode is the first byte of a multiple-byte instruction, additional  $\overline{INTA}$  pulses will be provided by the 8085A to clock in the remaining bytes. RESTART and CALL instructions are the most

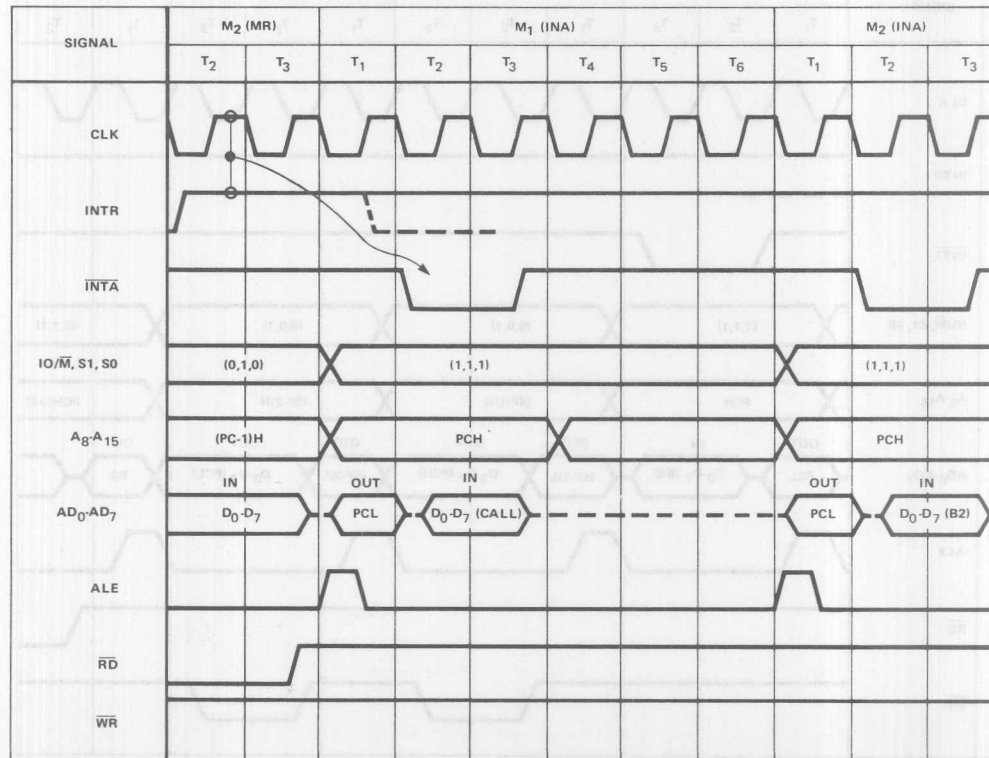


FIGURE 2-17 INTERRUPT ACKNOWLEDGE MACHINE CYCLES (WITH CALL INSTRUCTION IN RESPONSE TO INTR)



## FUNCTIONAL DESCRIPTION

logical choices, since they both force the processor to push the contents of the program counter onto the stack before jumping to a new location. In Figure 2-17 it is assumed that a CALL opcode is sent to the CPU during M<sub>1</sub>. The CALL opcode could have been placed there by a device like the 8259 programmable interrupt controller.

After receiving the opcode, the processor then decodes it and determines, in this case, that the CALL instruction requires two more bytes. The CPU therefore performs a second INA cycle (M<sub>2</sub>) to access the second byte of the instruction from the 8259. The timing of this cycle is identical to M<sub>1</sub>, except that it has only three T states. M<sub>2</sub> is followed by another INA cycle (M<sub>3</sub>) to access the third byte of the CALL instruction from the 8259.

Now that the CPU has accessed the entire instruction used to acknowledge the interrupt, it will execute that instruction. Note that any instruction could be used (except EI or DI, the instructions which enable or disable interrupts), but the RESTART and CALL instructions are the most logical choices. Also notice that the CPU inhibited the incrementing of the program counter (PC) during the three INA cycles, so that the correct PC value can be pushed onto the stack during M<sub>4</sub> and M<sub>5</sub>.

During M<sub>4</sub> and M<sub>5</sub>, the CPU performs MEMORY WRITE machine cycles to write the upper and then lower bytes of the PC onto the top of the stack. The CPU then places the two bytes accessed in M<sub>2</sub> and M<sub>3</sub> into the lower and upper bytes of the PC. This has the effect of jumping the execution of the program to the location specified by the CALL instruction.

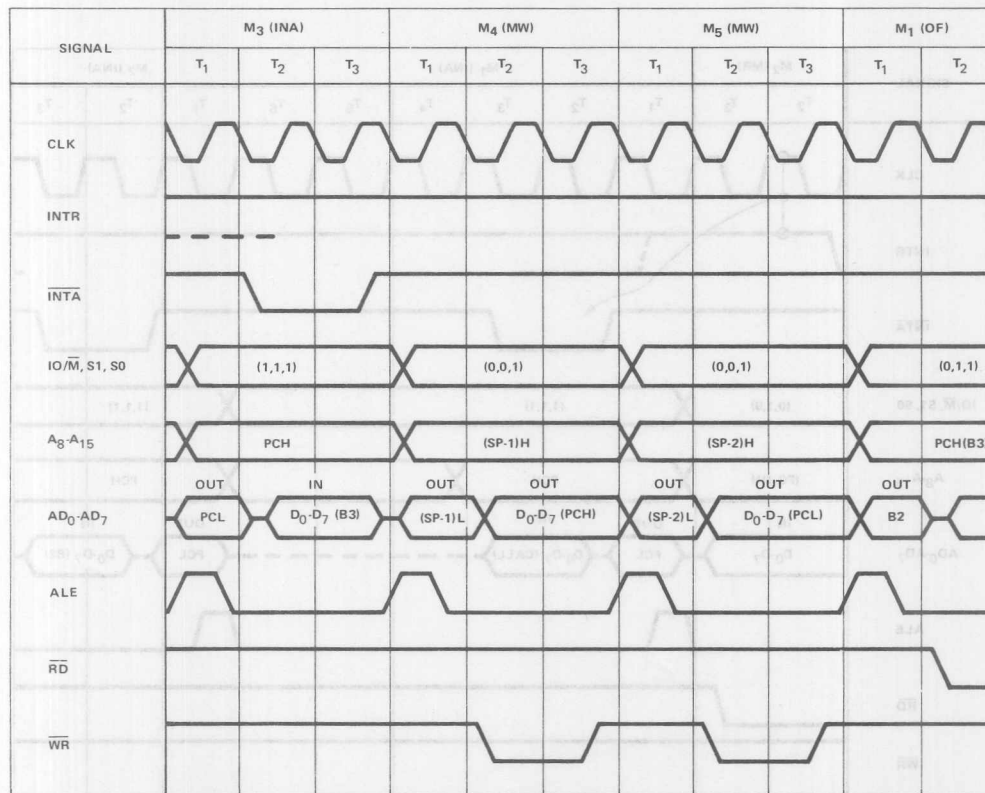


FIGURE 2-18 INTERRUPT ACKNOWLEDGE MACHINE CYCLES (WITH CALL INSTRUCTION IN RESPONSE TO INTR)

## 2.3.5 Bus Idle (BI) and HALT State

Most machine cycles of the 8085A are associated with either a READ or WRITE operation. There are two exceptions to this rule. The first exception takes place during  $M_2$  and  $M_3$  of the DAD instruction. The 8085A requires six internal T states to execute a DAD instruction, but it is not desirable to have  $M_1$  be ten (four normal plus six extra) states long. Therefore, the CPU generates two extra machine cycles that do not access either the memory or the I/O. These cycles are referred to as BUS IDLE (BI) machine cycles. In the case of DAD, they are identical to MR cycles except that RD remains high and ALE is not generated. Note that READY is ignored during  $M_2$  and  $M_3$  of DAD.

The other time when the BUS IDLE machine cycle occurs is during the internal opcode generation for the RST or TRAP interrupts. Figure 2-19 illustrates the BI cycle generated in response to RST 7.5. Since this interrupt is rising-edge-triggered, it sets an internal latch; that latch is sampled at the falling edge of the next to the last T-state of the previous instruction. At this point the CPU must generate its own internal RESTART instruction which will (in subsequent machine cycles) cause the processor to push the program counter on the stack and to vector to location 3CH. To do this, it executes an OF machine cycle without issuing RD, generating the RESTART opcode instead. After  $M_1$ , the CPU continues execution normally in all respects except that the state of the READY line is ignored during the BI cycle.

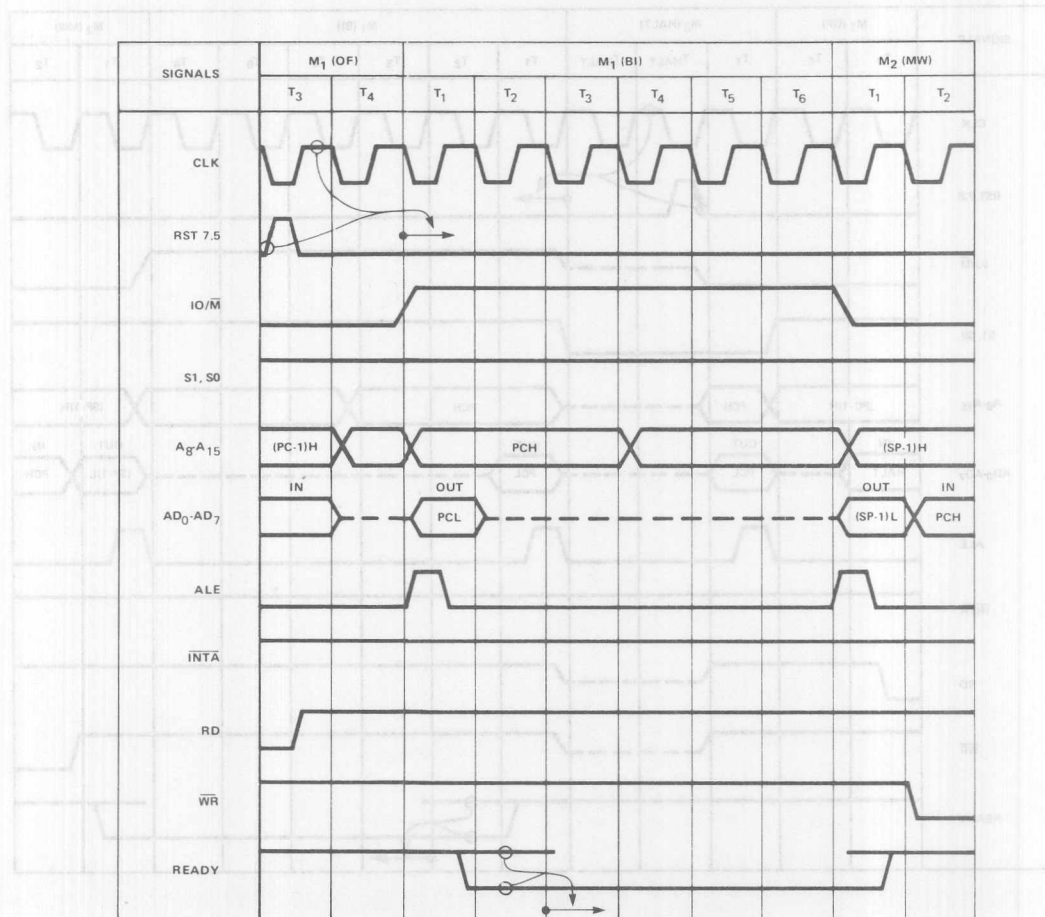


FIGURE 2-19 RST 7.5 BUS IDLE MACHINE CYCLE

has just been executed and the CPU is in the  $T_{\text{HALT}}$  state, with its various signals floating. There are only two ways the processor can completely exit the  $T_{\text{HALT}}$  state, as shown in Figure 2-11. The first way is for RESET to occur, which always forces the 8085A to  $T_{\text{RESET}}$ . The second way to exit  $T_{\text{HALT}}$  permanently is for a valid interrupt to occur, which will cause the CPU to disable further interrupts by resetting  $\text{INTE FF}$ , and to then proceed to  $M_1 \cdot T_1$  of the next instruction. When the HOLD input is activated, the CPU will exit  $T_{\text{HALT}}$  for the duration of  $T_{\text{HOLD}}$  and then return to  $T_{\text{HALT}}$ .

interrupt, it will set an internal latch which is sampled during  $\text{CLK} = "1"$  of every  $T_{\text{HALT}}$  state (as well as during  $\text{CLK} = "1"$  two  $T$  states before any  $M_1 \cdot T_1$ .) The fact that the latched interrupt was high (assuming that  $\text{INTE FF} = 1$  and the  $\text{RST 7.5 mask} = 0$ ) will force the CPU to exit the  $T_{\text{HALT}}$  state at the end of the next CLK period, and to enter  $M_1 \cdot T_1$ .

This completes our analysis of the timing of each of the seven types of machine cycles.

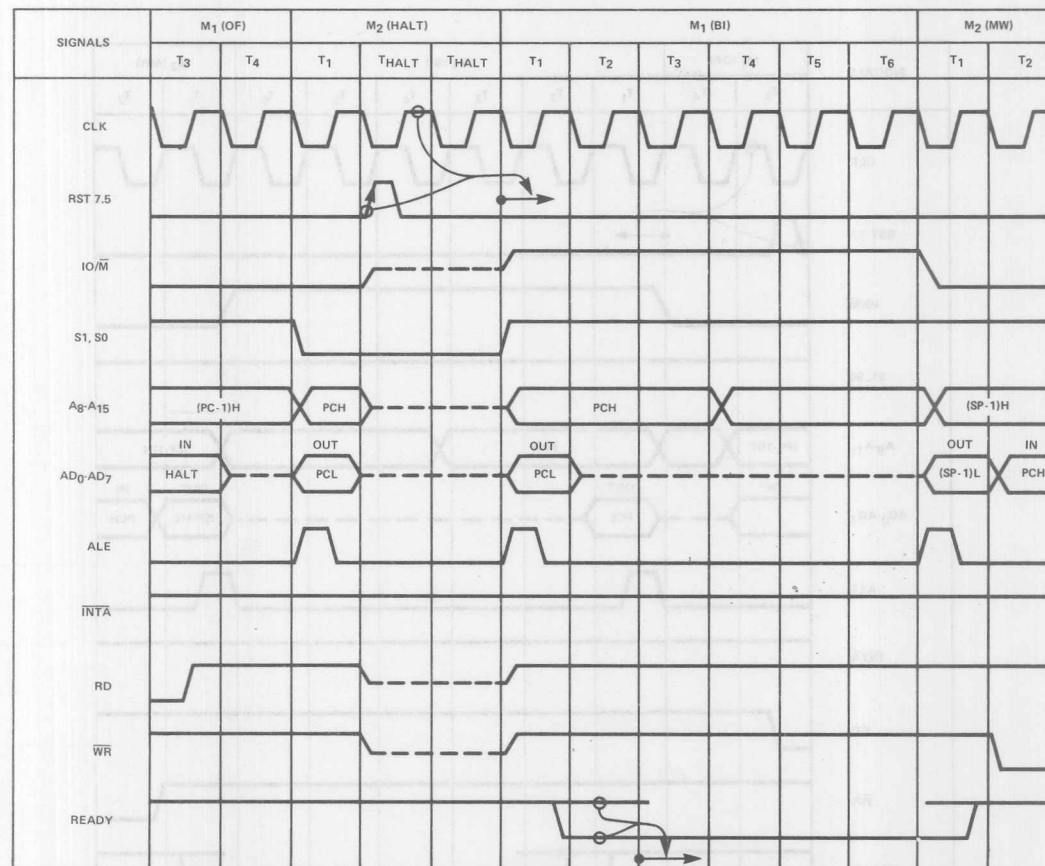


FIGURE 2-20 HALT STATE AND BUS IDLE MACHINE CYCLE  
RST 7.5 TERMINATES  $T_{\text{HALT}}$  STATE

## FUNCTIONAL DESCRIPTION

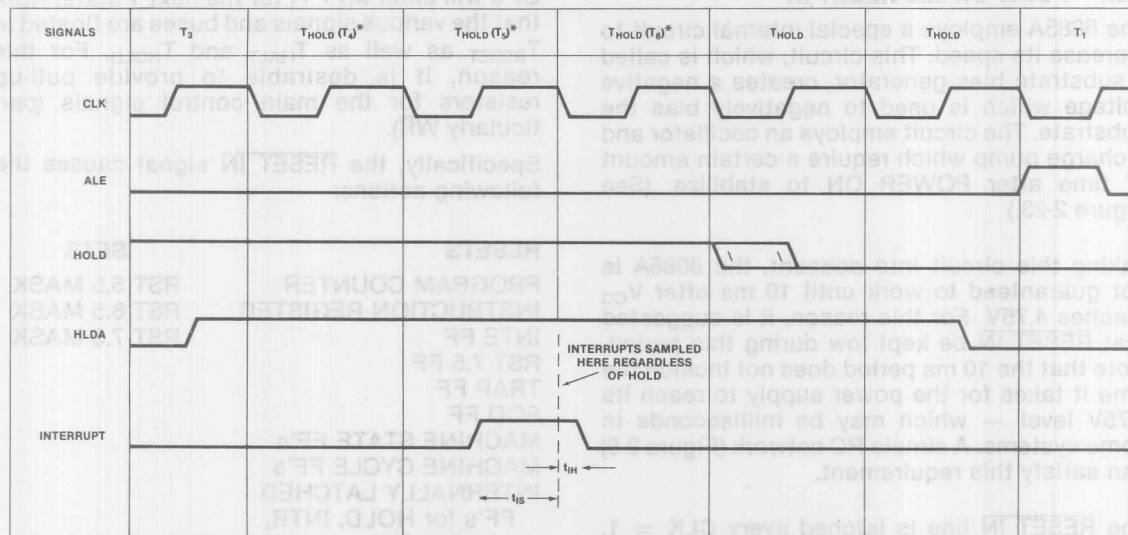
### 2.3.6 HOLD and HALT States

The 8085A uses the  $T_{\text{HOLD}}$  state to momentarily cease executing machine cycles, allowing external devices to gain control of the bus and perform DMA cycles. The processor internally latches the state of the HOLD line and the unmasked interrupts during  $\text{CLK} = "1"$  of every  $T_{\text{HOLD}}$  state. If the internal latched HOLD signal is high during  $\text{CLK} = "1"$  of any  $T_{\text{HOLD}}$  state, the CPU will exit  $T_{\text{HOLD}}$  and enter  $T_{\text{HOLD}}$  on the following  $\text{CLK} = "1"$ . As shown in Figure 2-21 this will occur even if a valid interrupt occurs simultaneously with the HOLD signal.

The state of the HOLD and the unmasked interrupt lines is latched internally during  $\text{CLK} = 1$  of each  $T_{\text{HOLD}}$  state as well as during  $T_{\text{HOLD}}$  states. If the internal latched HOLD signal is low during  $\text{CLK} = 1$ , the CPU will exit  $T_{\text{HOLD}}$  and enter  $T_{\text{HOLD}}$  on the following  $\text{CLK} = 1$ .

The 8085A accepts the first unmasked, enabled interrupt sampled; thereafter, all interrupt sampling is inhibited. The interrupt thus accepted will inevitably be executed when the CPU exits the HOLD state, even at the expense of holding off higher-priority interrupts (including TRAP). (See Figure 2-22.)

When the CPU is not in  $T_{\text{HOLD}}$  or  $T_{\text{HOLD}}$ , it internally latches the HOLD line only during  $\text{CLK} = 1$  of the last state before  $T_3$  ( $T_2$  or  $T_{\text{WAIT}}$ ) and during  $\text{CLK} = 1$  of the last state before  $T_5$  ( $T_4$  of a six T-state  $M_1$ ). If the internal latched HOLD signal is high during the next  $\text{CLK} = 1$ , the CPU will enter  $T_{\text{HOLD}}$  after the following clock. When the CPU is not in  $T_{\text{HOLD}}$  or  $T_{\text{HOLD}}$ , it will internally latch the state of the unmasked interrupts only during  $\text{CLK}$  of the next to the last state before each  $M_1 \cdot T_1$ .



\* SIGNIFIES THAT  $T_d, T_j, T_f$  MAY TAKE PLACE INSIDE THE 8085A EVEN WHILE THE PROCESSOR IS IN A HOLD STATE.

START OF INTERRUPT  
CYCLE DELAYED  
BY HOLD

FIGURE 2-21 HOLD VS INTERRUPT — NON HALT

## FUNCTIONAL DESCRIPTION

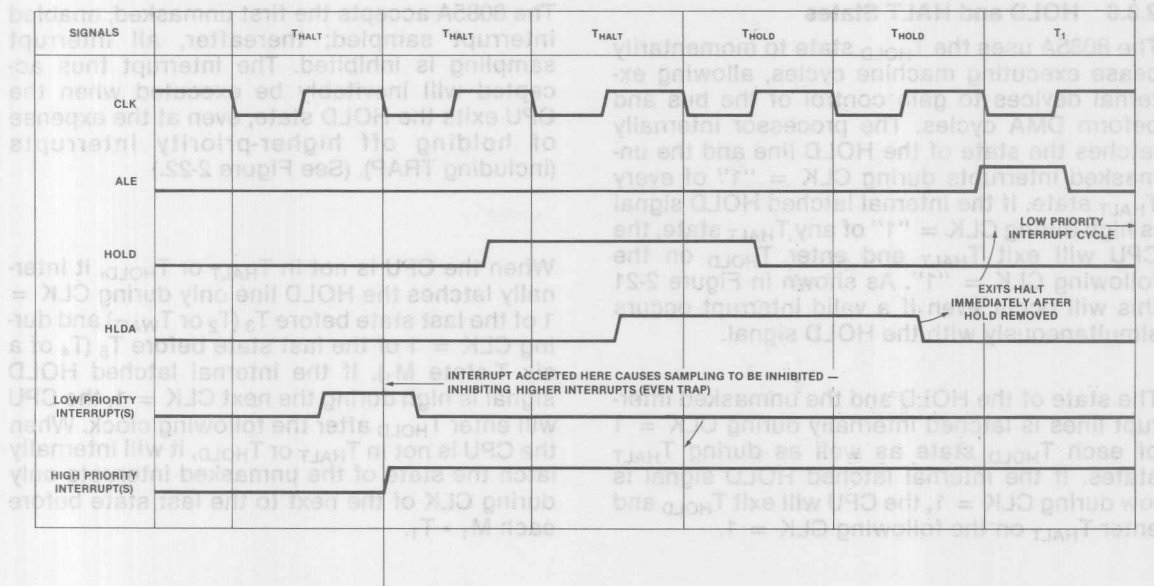


FIGURE 2-22 8085A HOLD VS INTERRUPTS — HALT MODE

### 2.3.7 Power On and RESET IN

The 8085A employs a special internal circuit to increase its speed. This circuit, which is called a substrate bias generator, creates a negative voltage which is used to negatively bias the substrate. The circuit employs an oscillator and a charge pump which require a certain amount of time after POWER ON to stabilize. (See Figure 2-23.)

Taking this circuit into account, the 8085A is not guaranteed to work until 10 ms after  $V_{CC}$  reaches 4.75V. For this reason, it is suggested that RESET IN be kept low during this period. Note that the 10 ms period does not include the time it takes for the power supply to reach its 4.75V level — which may be milliseconds in some systems. A simple RC network (Figure 3-6) can satisfy this requirement.

The RESET IN line is latched every CLK = 1. This latched signal is recognized by the CPU during CLK = 1 of the next T state. (See Figure 2-24.) If it is low, the CPU will issue RESET OUT and enter  $T_{HALT}$  for the next T state. RESET IN should be kept low for a minimum of three clock periods to ensure proper synchronization of the CPU. When the RESET IN signal goes high, the

CPU will enter  $M_1 \cdot T_1$  for the next T state. Note that the various signals and buses are floated in  $T_{RESET}$  as well as  $T_{HALT}$  and  $T_{HOLD}$ . For this reason, it is desirable to provide pull-up resistors for the main control signals (particularly  $\overline{WR}$ ).

Specifically, the RESET IN signal causes the following actions:

#### RESETS

PROGRAM COUNTER  
INSTRUCTION REGISTER  
INTE FF  
RST 7.5 FF  
TRAP FF  
SOD FF  
MACHINE STATE FF's  
MACHINE CYCLE FF's  
INTERNALLY LATCHED  
FF's for HOLD, INTR,  
and READY

#### SETS

RST 5.5 MASK  
RST 6.5 MASK  
RST 7.5 MASK

RESET IN does not explicitly change the contents of the 8085A registers (A, B, C, D, E, H, L) and the condition flags, but due to RESET IN occurring at a random time during instruction execution, the results are indeterminate.



## FUNCTIONAL DESCRIPTION

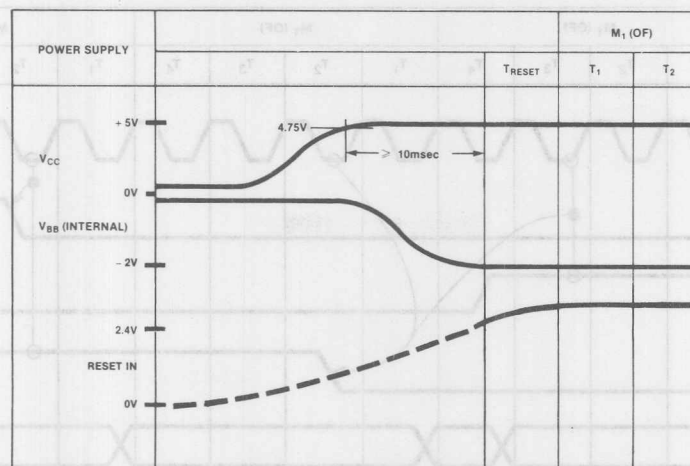


FIGURE 2-23 POWER-ON TIMING

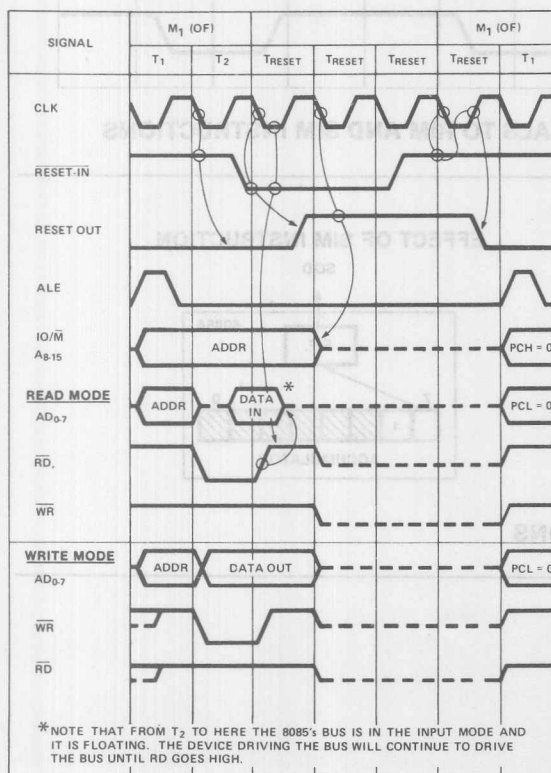


FIGURE 2-24 RESET IN TIMING

Following RESET, the 8085A will start executing instructions at location 0 with the interrupt system disabled, as shown in Figure 2-24.

Figure 2-24 also shows READ and WRITE operations being terminated by a RESET signal. Note that a RESET may prematurely terminate any READ or WRITE operation in process when the RESET occurs.

### 2.3.8 SID and SOD Signals:

Figure 2-25 shows the timing relationship of the SID and SOD signals to the RIM and SIM instructions. The 8085A has the ability to read the SID line into the accumulator bit 7 using RIM instructions. The state of the SID line is latched internally during T<sub>3</sub> • CLK = 0 of the RIM instruction. Following this, the state of the interrupt pins and masks are also transferred directly to the accumulator.

The 8085A can set the SOD flip-flop from bit 7 of the accumulator using the SIM instruction. (See Figure 2-26.) The data is transferred from the accumulator bit 7 to SOD during M<sub>1</sub> • T<sub>2</sub> • CLK = 0 of the instruction following SIM, assuming that accumulator bit 6 is a 1. Accumulator bit 6 is a "serial output enable" bit.

## FUNCTIONAL DESCRIPTION

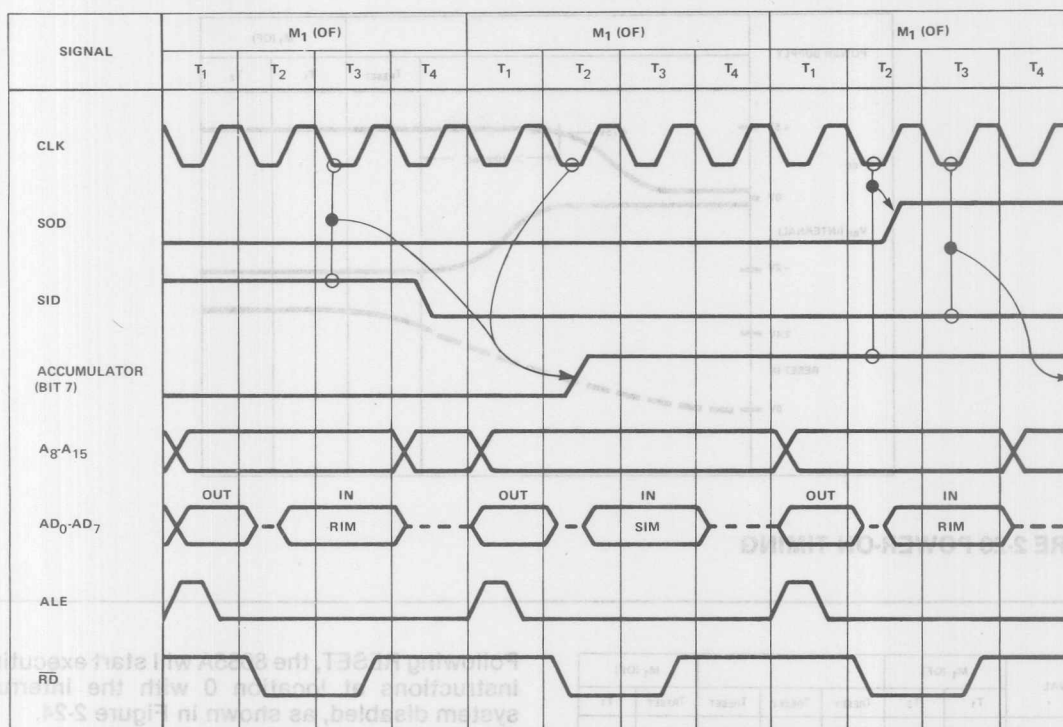
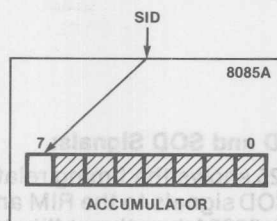


FIGURE 2-25 RELATIONSHIP OF SID AND SOD SIGNALS TO RIM AND SIM INSTRUCTIONS

### EFFECT OF RIM INSTRUCTION



### EFFECT OF SIM INSTRUCTION

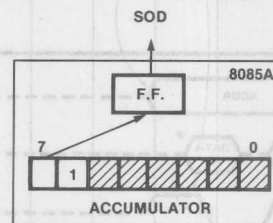


FIGURE 2-26 EFFECT OF RIM AND SIM INSTRUCTIONS

## 2.4 COMPARISON OF MCS-80 AND MCS-85 SYSTEM BUSES

This section compares the MCS-80 bus with the MCS-85 bus. Figure 2-28 details the signals and general timing of the two buses; the timing diagrams are drawn to the same scale (8080A clock cycle = 480 ns and 8085A clock cycle = 320 ns) to facilitate comparison.

## FUNCTIONAL DESCRIPTION

### MCS-80™ System Bus

The MCS-80 bus is terminated on one end by the CPU-GROUP (consisting of the 8080A, 8224, 8228) and on the other end by the various memory and I/O circuits. The following figure shows the major signals of the MCS-80 bus.

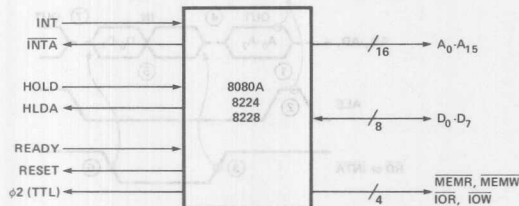


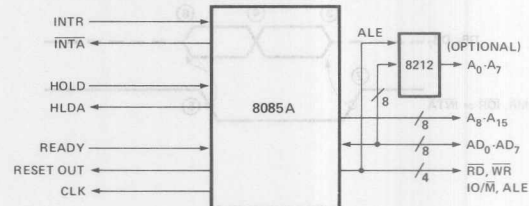
FIGURE 2-27 COMPARISON OF SYSTEM BUSES

### MCS-80™ System Bus

SIGNAL(S)	FUNCTION
A <sub>0</sub> -A <sub>15</sub>	The 16 lines of the address bus identify a memory or I/O location for a data transfer operation.
D <sub>0</sub> -D <sub>7</sub>	The 8 lines of the data bus are used for the parallel transfer of data between two devices.
MEMR, MEMW, IOR, IOW, INTA	These five control lines (MEMORY READ, MEMORY WRITE, I/O READ, I/O WRITE, and INTERRUPT ACKNOWLEDGE) identify the type and timing of a data transfer operation.
READY, RESET, HOLD, HLDA, φ2 (TTL), INT	These signals are used for the synchronization of slow speed memories, system reset, DMA, system timing, and CPU interrupt.

### MCS-85™ System Bus

The MCS-85 bus is terminated on one end by the 8085A and the other end by various memory and I/O devices. The MCS-85 bus may be optionally de-multiplexed with an 8212 eight bit latch to provide an MCS-80 type bus. The following figure shows the major signals of the MCS-85 bus.



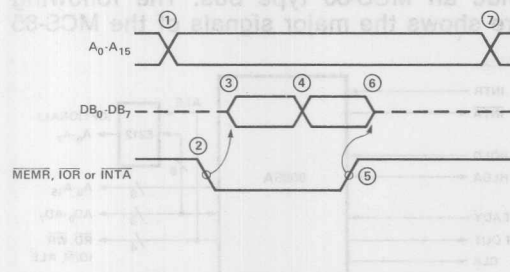
### MCS-85™ System Bus

SIGNAL(S)	FUNCTION
A <sub>8</sub> -A <sub>15</sub>	These are the high order eight bits of the address, and are used to identify a memory or I/O location for a data transfer cycle.
AD <sub>0</sub> -AD <sub>7</sub>	These eight lines serve a dual function. During the beginning of a data transfer operation, these lines carry the low order eight bits of the address bus. During the remainder of the cycle, these lines are used for the parallel transfer of data between two devices.
RD, WR, INTA	These signals identify the type and timing of a data transfer cycle.
I/O/M	The I/O/MEMORY line identifies a data transfer as being in the I/O address space or the memory address space.
ALE	ADDRESS LATCH ENABLE enables the latching of the A <sub>0</sub> -A <sub>7</sub> signals.
READY, RESET OUT, HOLD, HLDA, CLK, INTR	These signals are used for the synchronization of slow speed memories, system reset, DMA, system timing and CPU interrupt.

FIGURE 2-28 COMPARISON OF SYSTEM BUSES

## MCS-80™ System Bus for READ CYCLE

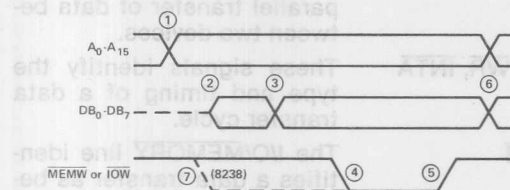
The basic timing of the MCS-80 BUS for a READ CYCLE is as follows:



The MCS-80 first presents the address ① and shortly thereafter the control signal ②. The data bus, which was in the high impedance state, is driven by the selected device ③. The selected device eventually presents the valid data to the processor ④. The processor raises the control signal ⑤, which causes the selected device to put the data bus in the high impedance state ⑥. The processor then changes the address ⑦ for the start of the next data transfer.

## MCS-80™ System Bus for WRITE CYCLE

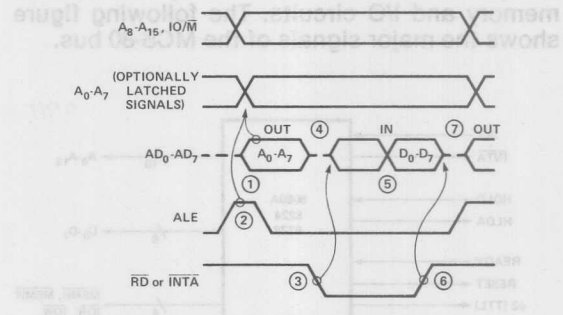
The basic timing of the MCS-80 BUS for a WRITE CYCLE is as follows:



The MCS-80 first presents the address ①, then enables the data bus driver ②, and later presents the data ③. Shortly thereafter, the MCS-80 drops the control signal ④ for an interval of time and then raises the signal ⑤. The MCS-80 then changes the address ⑥ in preparation for the next data transfer. The advance write signal of the 8238 is also shown ⑦.

## MCS-85™ System Bus for READ CYCLE

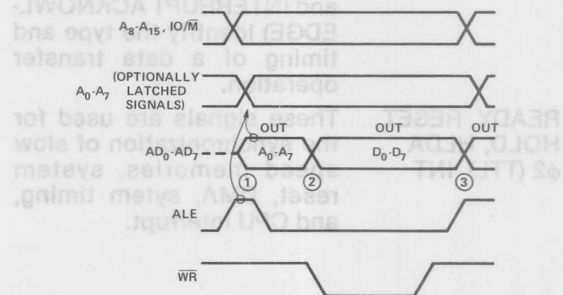
The basic timing of the MCS-85 BUS for a READ CYCLE is as follows:



At the beginning of the READ cycle, the 8085A sends out all 16 bits of address ①. This is followed by  $ALE$  ② which causes the lower eight bits of address to be latched in either the 8155/56, 8355, 8755A, or in an external 8212.  $RD$  is then dropped ③ by the 8085A. The data bus is then tri-stated by the 8085A in preparation for the selected device driving the bus ④; the selected device will continue to drive the bus with valid data ⑤, until  $RD$  is raised ⑥ by the 8085A. At the end of the READ CYCLE ⑦, the address and data lines are changed in preparation for the next cycle.

## MCS-85™ System Bus for WRITE CYCLE

The basic timing of the MCS-85 BUS for a WRITE CYCLE is as follows:



The timing of the WRITE CYCLE is identical to the MCS-85 READ CYCLE with the exception of the  $AD_0-AD_7$  lines. At the beginning of the cycle ①, the low order eight bits of address are on  $AD_0-AD_7$ . After  $ALE$  drops, the eight bits of data ② are put on  $AD_0-AD_7$ . They are removed ③ at the end of the WRITE CYCLE, in anticipation of the next data transfer.

FIGURE 2-28 (Continued) COMPARISON OF SYSTEM BUSES

The following observations of the two buses can be made:

1. The access times from address leaving the processor to returning data are almost identical, even though the 8085A is operating 50% faster than the 8080.
2. With the addition of an 8212 latch to the 8085A, the basic timings of the two systems are very similar.
3. The 8085A has more time for address setup to  $\overline{RD}$  than the 8080.
4. The MCS-80 has a wider  $\overline{RD}$  signal, but a narrower  $\overline{WR}$  signal than the 8085A.
5. The MCS-80 provides stable data setup to the leading and trailing edges of  $\overline{WR}$ , while the 8085 provides stable data setup to only the trailing edge of  $\overline{WR}$ .
6. The MCS-80 control signals have different widths and occur at different points in the machine cycle, while the 8085A control signals have identical timing.
7. While not shown on the chart, the MCS-80 data and address hold times are adversely affected by the processor preparing to enter the HOLD state. The 8085A has identical timing regardless of entering HOLD.
8. Also not shown on the chart is the fact that all output signals of the 8085A have  $-400\mu\text{a}$  of source current and 2.0 ma of sink current. The 8085A also has input voltage levels of  $V_{IL} = 0.8\text{V}$  and  $V_{IH} = 2.0\text{V}$ .

## CONCLUSION:

The preceding discussion has clearly shown that the MCS-85 bus satisfies the two restrictions of COMPATIBILITY and SPEED. It is compatible because it requires only an 8212 latch to generate an MCS-80 type bus. If the four control signals  $\overline{MEMR}$ ,  $\overline{MEMW}$ ,  $\overline{IOR}$  and  $\overline{IOW}$  are desired, they can be generated from  $\overline{RD}$ ,  $\overline{WR}$ ,

and  $\text{IO}/\overline{M}$  with a decoder or a few gates. The MCS-85 bus is also fast. While running at 3MHz, the 8085A generates better timing signals than the MCS-80 does at 2MHz. Furthermore, the multiplexed bus structure doesn't slow the 8085A down, because it is using the internal states to overlap the fetch and execution portions of different machine cycles. Finally, the MCS-85 can be slowed down or sped up considerably, while still providing reasonable timing.

TO USE. The  $\overline{RD}$ ,  $\overline{WR}$ , and  $\overline{INTA}$  control signals all have identical timing, which isn't affected by the CPU preparing to enter the HOLD state. Furthermore, the address and data bus have good setup and hold times relative to the control signals. The voltage and current levels for the interface signals will all drive buses of up to 40 MOS devices, or 1 schottky TTL device.

The MCS-85 system bus is also EFFICIENT. Efficiency is the reason that the lower eight address lines are multiplexed with the data bus. Every chip that needs to use both  $A_0-A_7$  and  $D_0-D_7$  saves 7 pins (the eighth pin is used for ALE) on the interface to the processor. That means that 7 more pins per part are available to either add features to the part or to use a smaller package in some cases. In the three chip system shown in Figure 3-6, the use of the MCS-85 bus saves  $3 \times 7 = 21$  pins, which are used for extra I/O and interrupt lines. A further advantage of the MCS-85 bus is apparent in Figure 3-7, which shows a printed circuit layout of the circuit in Figure 3-6. The reduced number of pins and the fact that compatible pinouts were used, provides for an extremely compact, simple, and efficient printed circuit. Notice that great care was taken when the pinouts were assigned to ensure that the signals would flow easily from chip to chip to chip.





**Chapter 3**  
**8085A**  
**System**  
**Operation and**  
**Interfacing**

**MOS**

**8085**

**MOS**

**8085**



## CHAPTER 3

## 8085A SYSTEM OPERATION AND INTERFACING

## 3.1 INTERFACING TO THE 8085A

The 8085A interfaces to both memory and I/O devices by means of READ and WRITE machine cycles, the timing of which are identical. During each machine cycle the 8085A issues an address and a control signal, then either sends data out on the bus or reads data from the bus. The 8085A may be performing a READ machine cycle, but what it reads could be a ROM, RAM, I/O device, peripheral device, or nothing.

There is no distinction between data, instruction opcodes, and I/O port numbers except the way the CPU interprets what it reads from the bus. If an opcode is what would logically appear on the bus, the CPU will treat as an opcode whatever does appear there; if an I/O port number is to be expected, what appears will be interpreted as a port number. The same is true for a WRITE cycle. The 8085A issues an address, data, and a control signal. Unless it is requested to WAIT (by use of the READY line) it will complete the cycle and proceed to the next. Regardless of whether there is a device present to accept the data, the CPU executes one instruction at a time, in sequence, until told to do otherwise. The program controls the sequence and nature of all machine cycles until an interrupt occurs.

There are two ways of addressing I/O devices in the MCS-85 system. If the IO/M output from the CPU is used to distinguish between I/O and memory READ and WRITE cycles, then that system is said to employ standard, or I/O-mapped, I/O. If IO/M is not so used, the CPU does not distinguish between I/O and memory, and its system employs memory-mapped I/O. Each method of addressing I/O has advantages and disadvantages.

## 3.2 MEMORY-MAPPED I/O

## 3.2.1 Advantages of Memory-Mapped I/O

Since the processor doesn't distinguish I/O from memory using this addressing scheme, you can take advantage of the larger instruction set that references the memory address space. Instead of only being able to transfer a byte of data between the accumulator and the I/O port (using INPUT and OUTPUT instructions), you can now program

arithmetic and logic operations on port data as well as move data between any internal register and the I/O port. Consider the new meaning of the following instructions:

Examples:

MOVr,M	(Input Port to any Register)
MOV M,r	(Output any Register to Port)
MVI M	(Output immediate data to Port)
LDA	(Input Port to ACC)
STA	(Output from ACC to Port)
LHLD	(16-Bit Input)
SHLD	(16-Bit Output)
ADD M	(Add Port to ACC)
ANA M	(AND Port with ACC)

## 3.2.2 Disadvantages of Memory-Mapped I/O

While memory instructions may increase the flexibility of the I/O system, there are some drawbacks. Since I/O devices are now addressed as memory, there are fewer addresses available for memory. A common practice is to use address bit 15 ( $A_{15}$ ) to distinguish memory from I/O. (See Figure 3-2 and accompanying discussion.) If  $A_{15}=0$  then memory is being addressed; if  $A_{15}=1$ , I/O is being addressed. This particular scheme limits the maximum amount of memory that can be used to 32k bytes. A further disadvantage of memory-mapped I/O is that it takes 3 bytes of instruction and 13 clock cycles using the LDA or STA instructions to specify moving a byte of data between the accumulator and an I/O device, whereas the INPUT and OUTPUT instructions require only two bytes and 10 clock cycles. This is because the I/O address space is smaller (only 256 bytes) and therefore requires fewer bits to completely specify an address. A further advantage of using the INPUT and OUTPUT Instructions is that it allows the easy connection of the MCS-80 peripherals to the MCS-85 multiplexed bus. If you memory-map the MCS-80 peripherals to the MCS-85 bus, you must either latch the lower address bits with an 8212 or use a portion of the memory address space by connecting the chip selects and address lines of the ports to the unmultiplexed upper eight lines of the address bus.

### 3.3 ADDRESS ASSIGNMENT

#### 3.3.1 Decoding

Besides memory-mapped I/O, another practice is to only partially decode the address bus when generating chip selects. Every device has a given number of unique addresses associated with it. The 8355, for instance, has 2k bytes of ROM and therefore has 2k addresses associated with the ROM. Any one of these 2k addresses can be uniquely specified by a pattern on the 11 ( $2^{11} = 2k$ ) address lines. However, since the 8355 must work with other devices in a system, it isn't enough to simply specify the 11 bits; further bits of information must be used to locate the 2k bytes within the 65k address space. The 2k bytes are located by the use of chip enable (CE) inputs to the 8355 chip. If the 8355 were to occupy the first 2k bytes of the memory address space, it would, strictly speaking, be necessary to decode the fact that  $A_{15}-A_{11}$  were all zeroes, and use that condition as a chip enable. Then the 8355 would be selected only when the address bus was less than 2k.

However, if other 2k blocks of addresses aren't being used, you may combine those addresses and not decode all of the upper five address lines for chip enables. In fact, in a small system you may need to decode only one bit of address, which is to say connect that bit of the address bus to the chip enable line of the 8355. If you connect  $A_{11}$  to the  $\overline{CE}$  line of the 8355 and tie CE to  $V_{CC}$ , then the 8355 would be selected whenever the memory address was less than 2k. (See Figure 3-1A.)

However, it will also be selected whenever memory locations 4k-6k, 8k-10k, 61k-63k (i.e., whenever bit  $A_{11} = 0$ ) is addressed. If the programmer is aware of this, and if there are no other devices assigned to the other address spaces, then it may be an acceptable condition. Care must be taken, however, to ensure that at no time will two different devices be selected simultaneously. Whenever one device is selected, that memory address must deselect all other devices. If two devices are selected simultaneously for a READ operation, the electrical conflict on the bus may damage one or both parts. Note also that the address bus may reflect an undesired address during  $T_5$ ,  $T_6$  of an opcode fetch cycle and during address bus transitional periods in  $T_1$  (this is illustrated in Chapter 2). Therefore, all memory and I/O devices must qualify their selection with  $\overline{RD}$  or  $\overline{WR}$ , or the address on the bus at the falling edge of the ALE, so as to ignore all spurious addresses.

#### 3.3.2 Linear Selection

Using an address bit as a chip select is referred to as linear selection. The direct consequence of linear selection is that you cut the available address space in half for each single address bit used as a chip enable. If this penalty is too high, you can always use an 8205 one-of-eight decoder. Also, some chips have multiple chip enables, which allows for some automatic decoding of the address. (See Figures 3-1B and 3-1C.)

One drawback to linear selection is that the memory addresses of the different parts are not contiguous. For example, if three 8355s are addressed using linear selection, one might be located at 0-2k, the next at 6k-8k, and the next at 10k-12k. The programmer must recognize these page boundaries and jump over them.

### 3.4 INTERFACING TO THE 8155/8156, 8355/8755A

#### 3.4.1 I/O Mapped I/O:

This section describes some of the techniques involved in connecting the MCS-85 combination memory and I/O chips to the 8085A as I/O devices.

Figure 3.1A shows one 8355 connected to the 8085A bus. (In the interest of simplicity, only the chip enable and IO/ $\overline{M}$  lines are shown; the other lines are connected as shown in Figures 3.6, 3.7 or 3.8.) Notice that CE is tied to  $V_{CC}$  and  $\overline{CE}$  is connected to  $A_{11}$ . This is because after RESET the processor always starts executing at location 0. Since the ROM normally contains the program, it must be selected when the address is all zeroes.

One consequence of the ROM being selected by an all-zero address is that the I/O ports on the chip will be selected only when  $A_{11} = 0$ . This is because the I/O ports and the memory have common chip enables, therefore forcing the selection conditions of one onto the other. Furthermore, since the IO/ $\overline{M}$  line of the chip is connected to the IO/ $\overline{M}$  line of the 8085A, the port has I/O mapped I/O. The I/O ports can be accessed only by use of the INPUT and OUTPUT instructions; since these are the only instructions that cause IO/ $\overline{M}$  to go high.

The boxes to the right of the chip in Figure 3.1A indicate the memory addresses and I/O Port numbers required to access the chip. As a result of the linear selection technique used, there are many "don't care" bits (marked by "X"s) in the address. While they don't affect the addressing of this device, they may affect other



## SYSTEM OPERATION

FIGURE 3-1A SINGLE CHIP

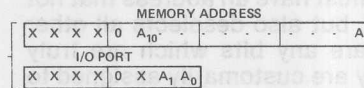
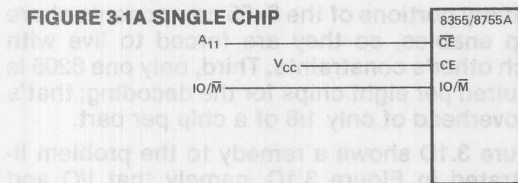


FIGURE 3-1B MULTIPLE CHIPS

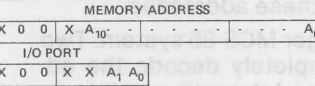
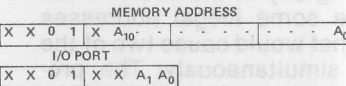
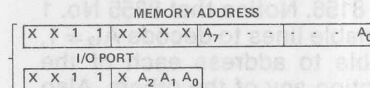
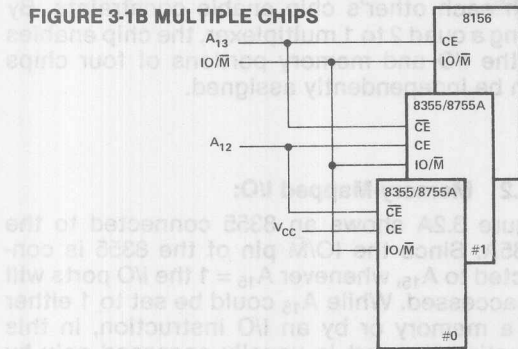


FIGURE 3-1C FULLY DECODED AND EXPANDED

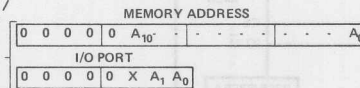
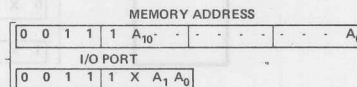
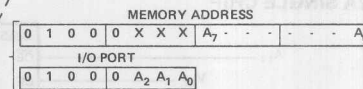
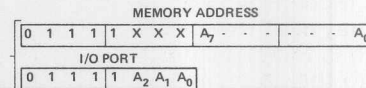
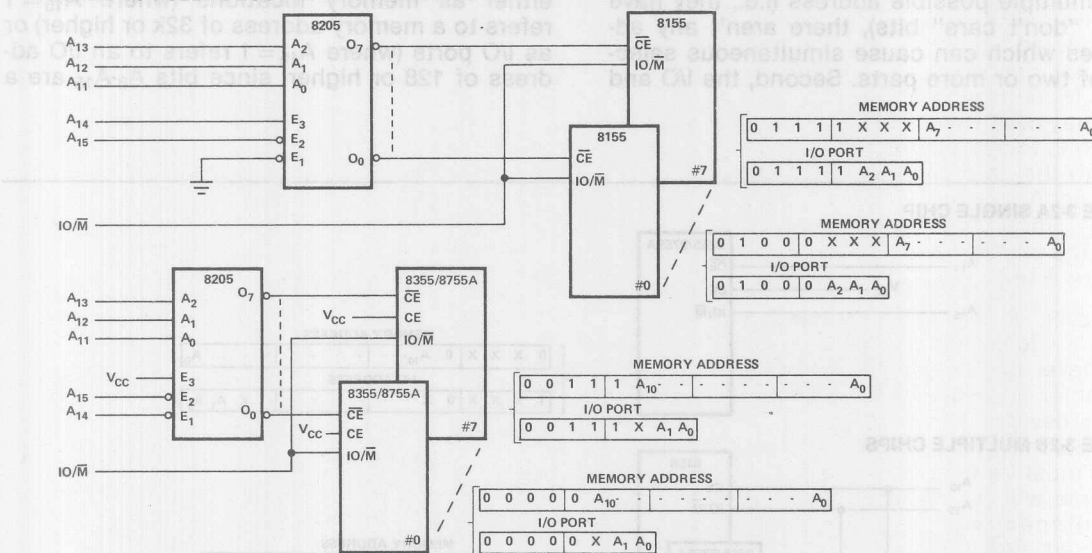


FIGURE 3-1D SEPARATE CHIP ENABLES FOR I/O AND MEMORY

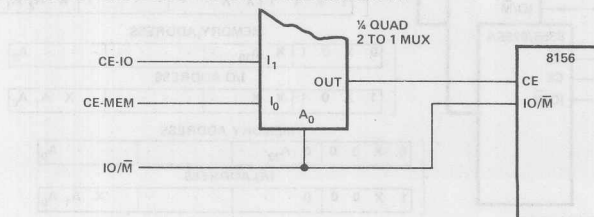


FIGURE 3-1 MCS-85™ PERIPHERALS WITH I/O MAPPED I/O

devices in the system, which would force them to be either ones or zeroes. Remember that two devices may not be selected simultaneously; thus each device must have an address that not only selects itself, but also deselects all other devices. If there are any bits which are truly "don't cares," they are customarily assigned to be zero. If all the "X" bits in Figure 3.1A were "don't cares," then the chip could be addressed as memory locations 0-2k, and I/O Ports 0-3.

Figure 3.1B shows a slightly larger system of two 8355s and one 8156. Notice that 8355 No. 1 uses its two chip enable lines to decode  $A_{12} = 1$ ,  $A_{13} = 0$ . It is possible to address each of the chips without selecting any of the others. Also notice that there are some illegal addresses (e.g.,  $A_{12} = 0$ ,  $A_{13} = 1$ ) that would cause two of the devices to turn on simultaneously. The programmer must not use these addresses.

Figure 3.1C shows a larger MCS-85 system. Two 8205s are used to completely decode the addresses. There are some interesting points to observe here. First, while some of the devices have multiple possible address (i.e., they have some "don't care" bits), there aren't any addresses which can cause simultaneous selection of two or more parts. Second, the I/O and

memory portions of the 8x55 components share chip enables, so they are forced to live with each other's constraints. Third, only one 8205 is required per eight chips for the decoding; that's an overhead of only 1/8 of a chip per part.

Figure 3.1D shows a remedy to the problem illustrated in Figure 3.1C, namely that I/O and memory portions of the chip are forced to live with each other's chip enable constraints. By using a quad 2 to 1 multiplexer, the chip enables of the I/O and memory portions of four chips can be independently assigned.

## 3.4.2 Memory-Mapped I/O:

Figure 3.2A shows an 8355 connected to the 8085A. Since the  $\overline{IO/\overline{M}}$  pin of the 8355 is connected to  $A_{15}$ , whenever  $A_{15} = 1$  the I/O ports will be accessed. While  $A_{15}$  could be set to 1 either by a memory or by an I/O instruction, in this situation the port is usually accessed only by the memory instructions. You may access ports either as memory locations (where  $A_{15} = 1$  refers to a memory address of 32k or higher) or as I/O ports (where  $A_{15} = 1$  refers to an I/O address of 128 or higher, since bits  $A_8-A_{15}$  are a

FIGURE 3-2A SINGLE CHIP

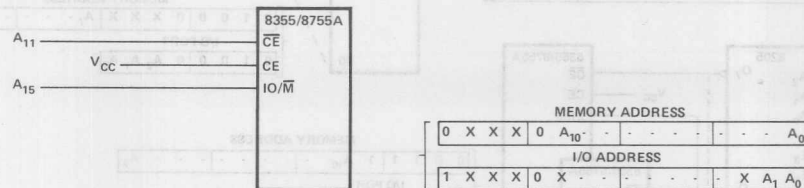


FIGURE 3-2B MULTIPLE CHIPS

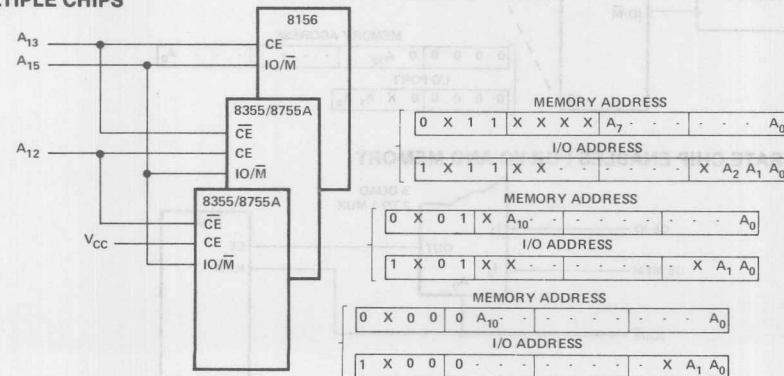


FIGURE 3-2 MCS-85™ PERIPHERALS WITH MEMORY-MAPPED I/O

replication of bits  $A_0-A_7$ ). Assuming that memory-mapped I/O is used, the addresses are shown in the boxes to the right in Figure 3-2. If you want to be sure that neither the I/O nor the memory is ever selected by any INPUT or OUTPUT instruction, then the chip enable must be conditioned by  $IO/\overline{M} = 0$ .

Figure 3.2B shows a somewhat larger system, also using memory-mapped I/O. As in Figure 3.1B care must be exercised to ensure that no two devices are accessed simultaneously. You can see that considerable memory address space is used up as a result of using memory-mapped I/O.

## 3.5 INTERFACING TO MCS-80™ PERIPHERALS

### 3.5.1 I/O Mapped I/O:

For want of a better name, the Intel® 825x, 827x, and 829x series peripherals are referred to here as MCS-80 peripherals because unlike the 8155/56, 8355 and 8755A, they are compatible with the nonmultiplexed MCS-80 system bus.

To interface to an MCS-80 peripheral, you must provide a constant address, a chip select, and  $\overline{RD}$  or  $\overline{WR}$ . Since the upper address lines ( $A_8-A_{15}$ ) of the 8085A are nonmultiplexed, they can be tied directly to the peripherals, as shown in Figure 3.3A. To provide I/O mapped I/O, use either linear selection (keeping the I/O and memory addresses noncoincident), or condition the chip selects  $\overline{WR}$  with  $IO/\overline{M} = 1$ . Figure 3.3A shows a technique of gating the chip selects with  $IO/\overline{M} = 1$ , using an 8205. This technique also allows more I/O devices to be used than linear selection would. Note that this technique relies on the fact that the I/O Port number is copied onto  $A_8-A_{15}$  as well as  $A_0-A_7$  during an INPUT or OUTPUT instruction.

Figure 3.3B shows an alternative approach to interfacing to MCS-80 components. By latching the lower 8 bits of address with an 8212, and decoding the control signals with an 8205, you create an exact copy of the MCS-80 (8080A, 8224, 8228) bus. You may then use whatever circuits have been previously developed for the 8080. The total cost is one 8212 and one 8205. Since the same signals might have needed buffering anyway (and the 8212 and 8205 provide buffering of their outputs), the extra component overhead ranges from little to nothing.

### 3.5.2 Memory-Mapped I/O:

Exactly the same techniques used to memory map the MCS-85 apply to the MCS-80 I/O devices. Figure 3.4 shows an 8205 used to qualify the chip select of the I/O device with  $IO/\overline{M} = 0$ . Since

the MCS-80 peripherals require nonmultiplexed address lines, linear select is not too useful unless the address lines are latched. This is because connecting both the chip selects and the address lines of the MCS-80 peripherals to  $A_8-A_{15}$  would deplete all the useful addresses very quickly.

## 3.6 INTERFACING TO STANDARD BUS MEMORIES

Standard bus memory devices are designed to be used with nonmultiplexed address and data buses. Interfacing to standard memories is very similar to interfacing to MCS-85 memories with the exception that  $A_0-A_7$  must be latched. Once this requirement is met, all the tricks discussed earlier can be used. Since the address lines would eventually require buffering as the system size grew, the overhead of the 8212 latch again becomes negligible.

Figure 3.5 shows the interface of the 8085A to a large block of memory, specifically 16k bytes of ROM and 8k bytes of RAM. Besides the memories, the circuit requires only 2-1/6 other parts for logical gating. If MCS-80 I/O parts were used, the 8212 latch could be shared between the two groups, further reducing the gating overhead per IC. Sixteen 2142 chips and eight 2316E chips are used in this design. The data bus, address lines 8-10, and control signals in this system all should be buffered. This applies to any system with the number of memory devices represented here.

Wherever two or more parts are paralleled on the same bus, they must be 3-state devices such as the 2142 RAM, 2316E ROM, 2716 EPROM, 2332 ROM, 2732 EPROM, and 2364 ROM, which have either an output disable (OD) input or multiple chip select (CS) inputs. To prevent bus contention, only one memory device may be output-enabled at a time in this configuration; the outputs of all others must be deselected during  $\overline{RD}$ .

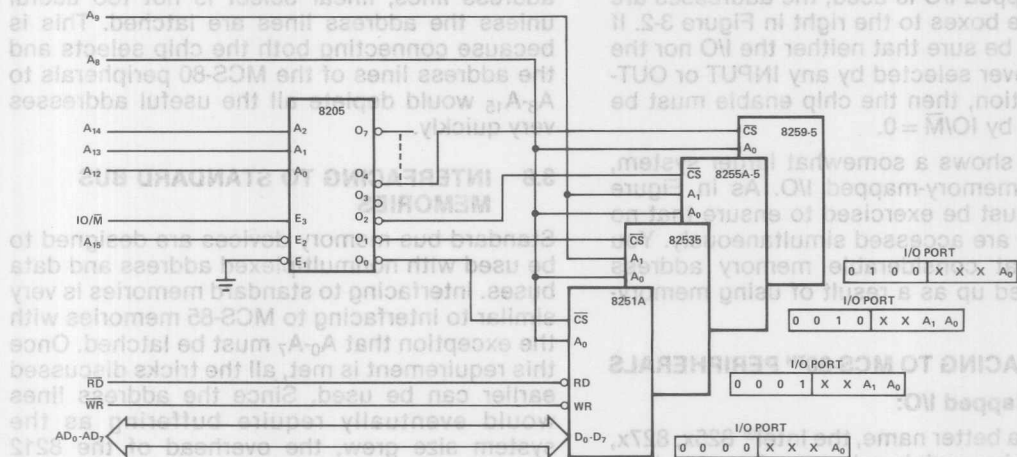
For additional information on interfacing standard memory devices, please read Section 2 of Appendix I and the Intel applications note AP-30 "Application of Intel's 5V EPROM and ROM Family for Microprocessor Systems" available from: Intel, Literature Dept., 3065 Bowers Ave., Santa Clara, CA 95051.

## 3.7 DYNAMIC RAM INTERFACE:

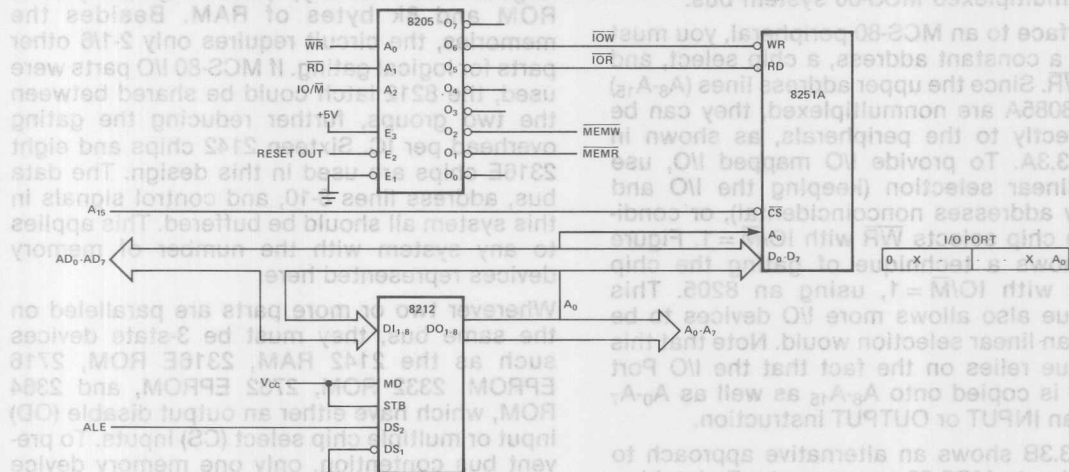
For interfacing the dynamic RAM, Intel makes a single-component dynamic RAM refresh controller, the 8202, which interfaces the 8085A to multiplexed-address-bus dynamic RAMs like

## SYSTEM OPERATION

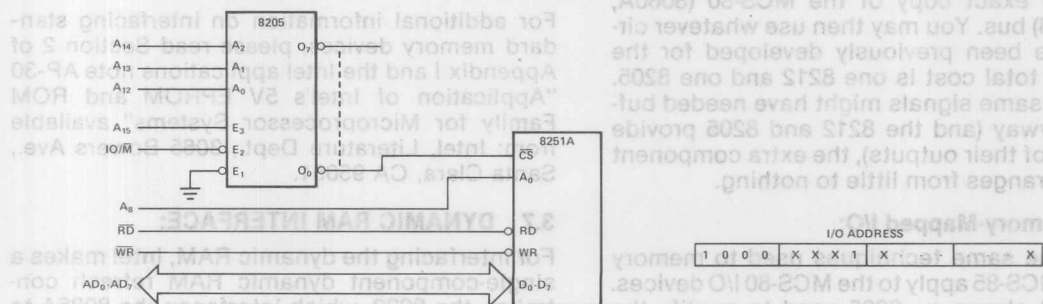
### FIGURE 3-3A DECODED CHIP SELECTS



**FIGURE 3-3B DECODED CONTROLS AND LATCHED ADDRESS (MCS-80™ TYPE BUS)**



### FIGURE 3-3 MCS-80™ PERIPHERALS WITH I/O MAPPED I/O



**FIGURE 3-4 MCS-80™ PERIPHERALS WITH MEMORY-MAPPED I/O AND DECODED CHIP SELECTS**

## SYSTEM OPERATION

the Intel 2104A and 2117. The 8202 provides the necessary refreshing for such dynamic RAMs, and also provides the control signals required for accessing, selecting, and address clocking. It allows for the use of the 8085A's full capability of 64k bytes of address space with no additional buffering devices. As with other standard memory interfaces, it is necessary to demultiplex the lower 8 bits of address from the multiplexed 8085A bus, AD<sub>0-7</sub>.

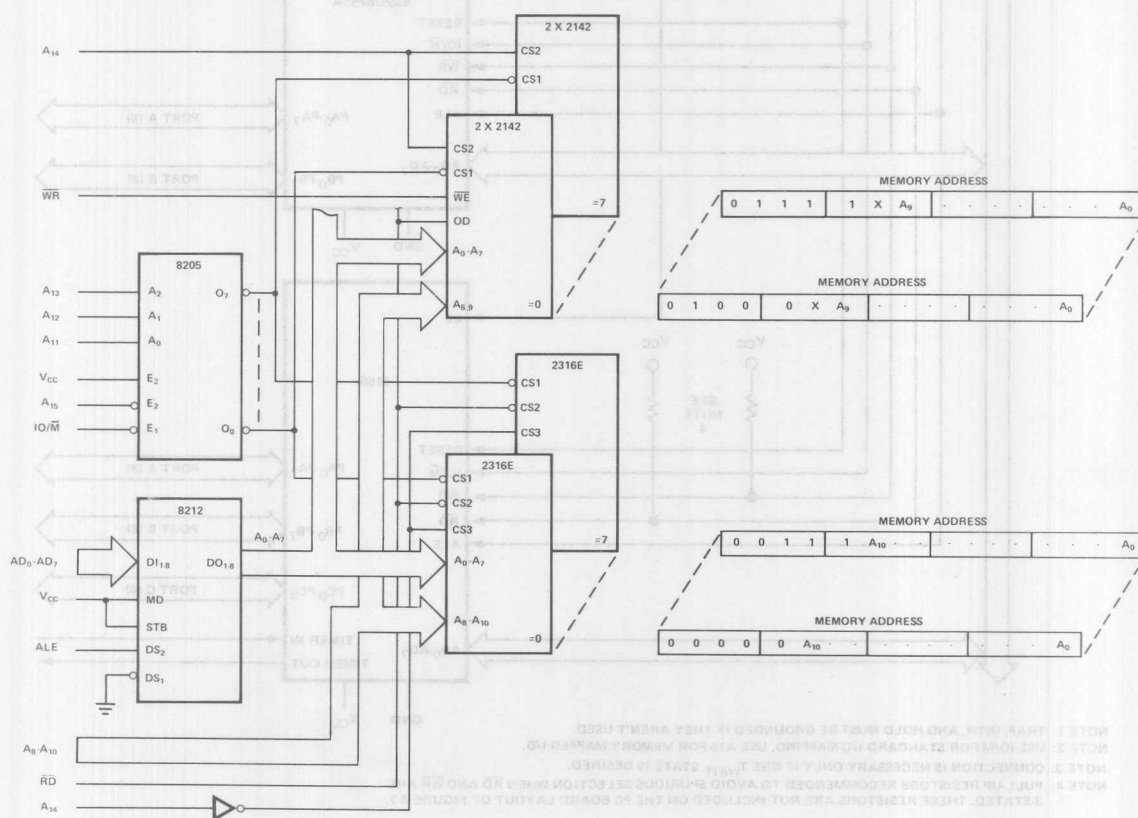
### 3.8 MINIMUM MCS-85™ SYSTEM

The Schematics of Figure 3.6 depict a minimum system core. In actual use, some of the processor control signals (TRAP, INTR, and HOLD) would have to be terminated. Also, interface logic to external devices as well as more memory and I/O devices may be desirable. The first thing one notices about the system in Figure 3.6 is the scarcity of parts required to build this system. With a minimum of parts, we

have constructed a microcomputer system that has the following functions:

PARTS	FUNCTIONS
1 8085A	1 CPU (Clock cycle $\leq 320$ ns)
1 8355/8755A	2048 Bytes of either EPROM or ROM
1 8156	256 Bytes of RAM
1 Crystal	38 I/O Lines
4 Resistors	5 Interrupts
1 Capacitor	1 Programmable Timer/Counter
1 Diode	1 Crystal and Oscillator
1 + 5 Power Supply	1 Clock
	1 Power-on Reset

By looking at the printed circuit layout of Figure 3.7, we can see that not only are there just 3 ICs, but that the interconnection of these parts is extremely easy and provides a very dense layout. Especially notice the easy flow of the system bus on the solder side of the board.



**FIGURE 3-5 STANDARD MEMORIES WITH LATCHED ADDRESS AND DECODED CHIP SELECTS**



## SYSTEM OPERATION

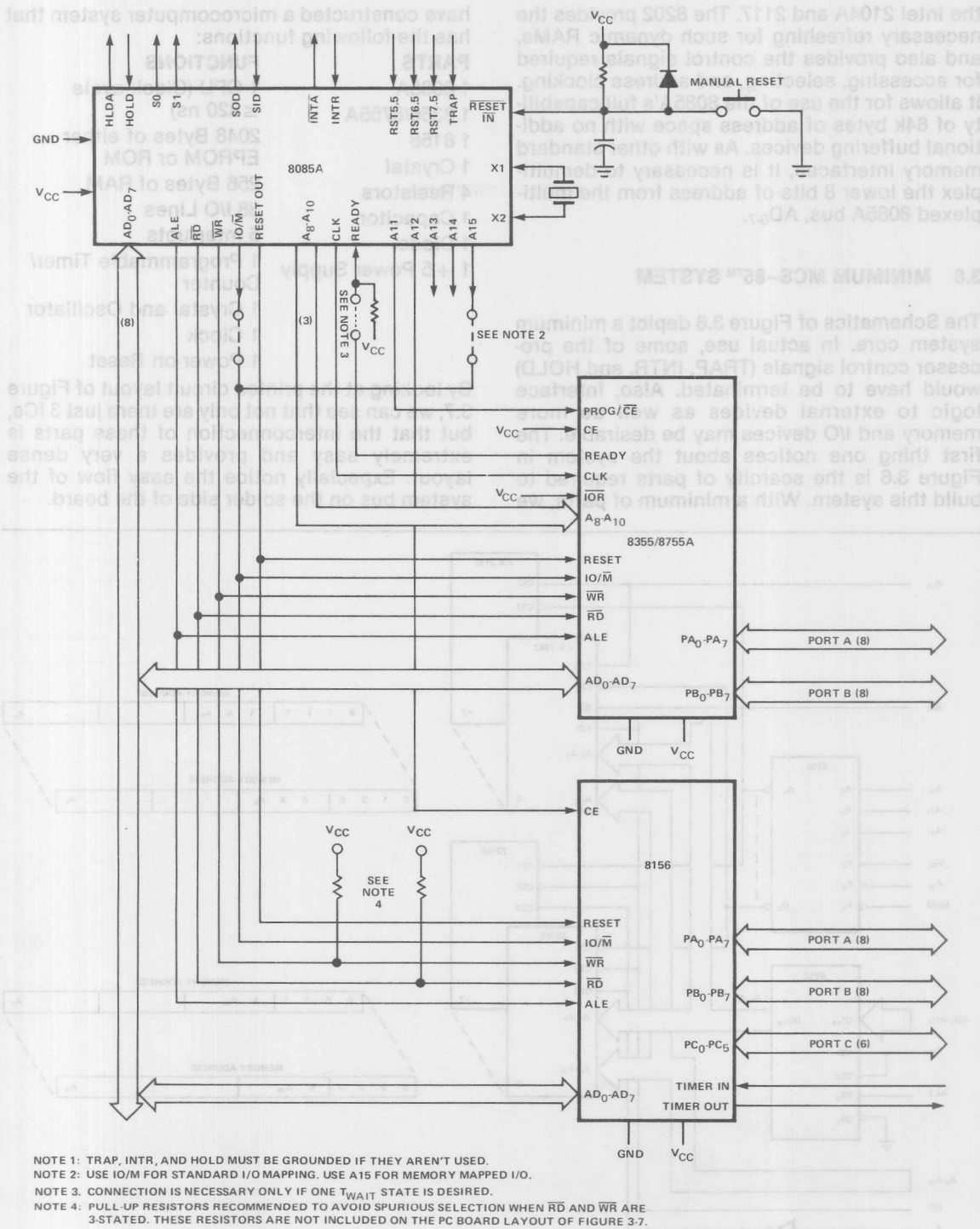


FIGURE 3-6 MINIMUM 8085 SYSTEM

# SYSTEM OPERATION

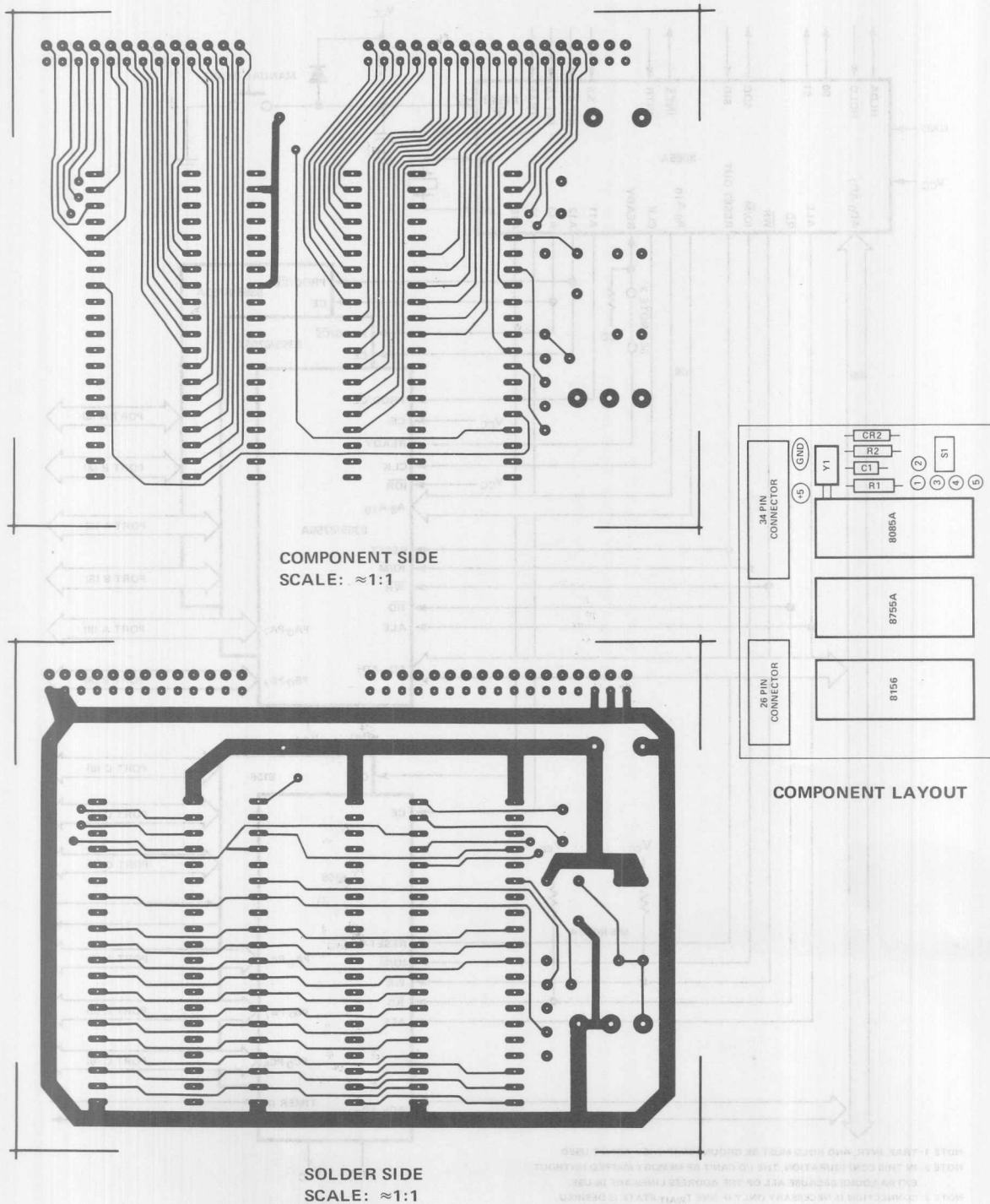
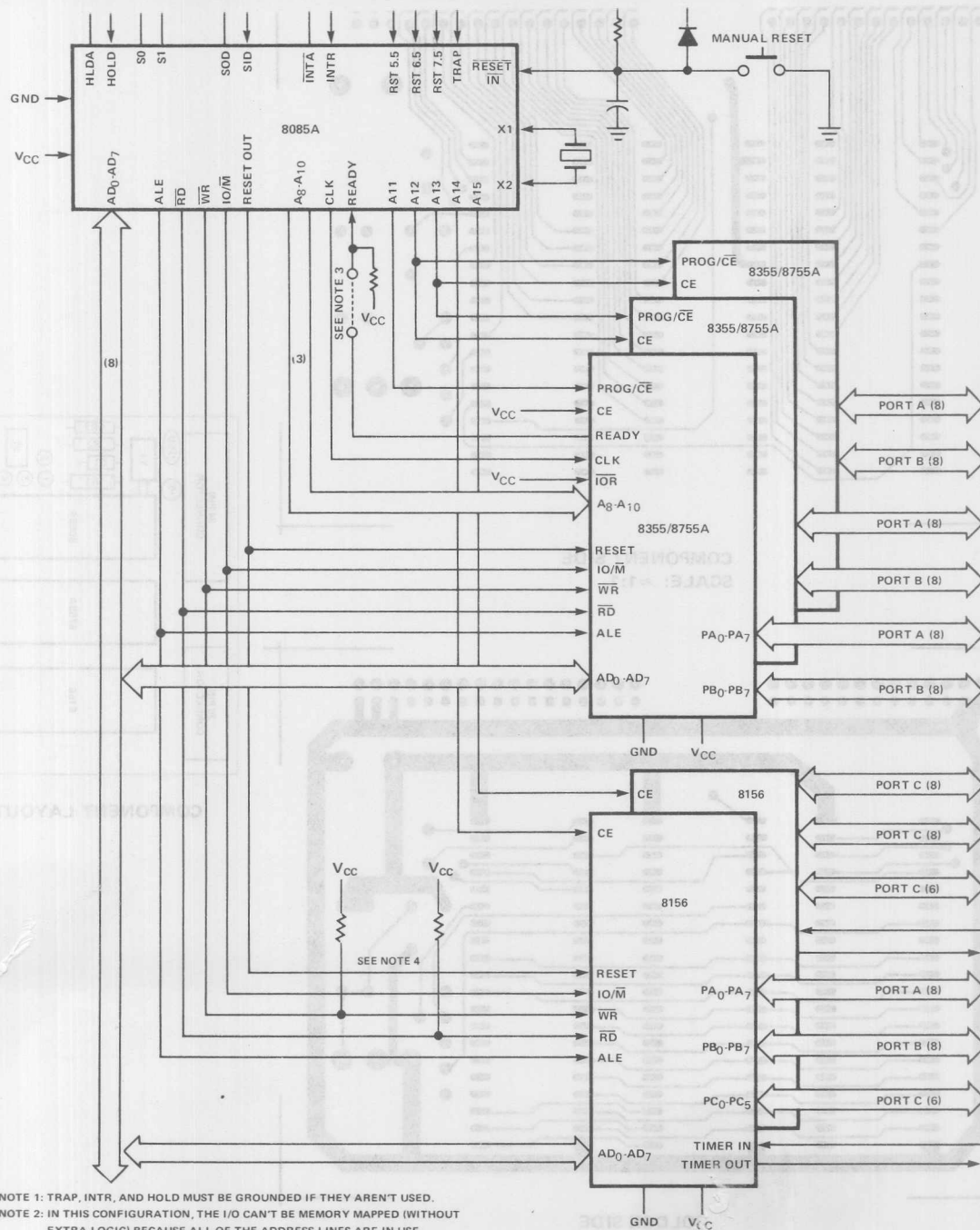


FIGURE 3-7 PRINTED CIRCUIT LAYOUT



**FIGURE 3-8 EXPANDED SYSTEM**

### 3.9 EXPANDED MCS-85™ SYSTEM

Figure 3.8 shows the circuit Figure 3.6 expanded to its maximum size without the use of any extra logic. In an extremely small board area we can fit:

#### PARTS

- 1 8085A
- 3 8355/8755A
- 2 8156
- 1 Crystal
- 4 Resistors
- 1 Capacitor
- 1 Diode

#### FUNCTION

- 1 CPU (Clock cycle  $\leq 320$ ns)
- ROM/EPROM 6144 Bytes
- 512 Bytes RAM
- 76 I/O Lines
- 5 Interrupts
- 2 Programmable Timer/Counters
- 2 Serial I/O Lines
- 1 Crystal and Oscillator
- 1 Clock
- 1 Power-on Reset

### 3.10 MCS-85™ SYSTEM WITH 8185

The 8185 1K-byte static RAM chip is another multiplexed-bus component that insures that the most highly integrated systems can be built with MCS-85 components. Figure 3.9 shows a 4-chip MCS-85 system schematic with the following characteristics:

#### PARTS

- 1 8085A
- 1 8185
- 1 8156
- 1 8355/8755A

#### FUNCTION

- 1 CPU
- ROM/EPROM 2048 Bytes
- 1280 Bytes RAM
- 38 I/O Lines
- 5 Interrupts
- 1 Timer/Counter
- 2 Serial I/O Lines

The 8185 also has power-down capability. By connecting  $\overline{CE}_1$  to  $IO/\overline{M}$  from the 8085A the 8185 will be powered down during I/O operations and Interrupt Acknowledge cycles.

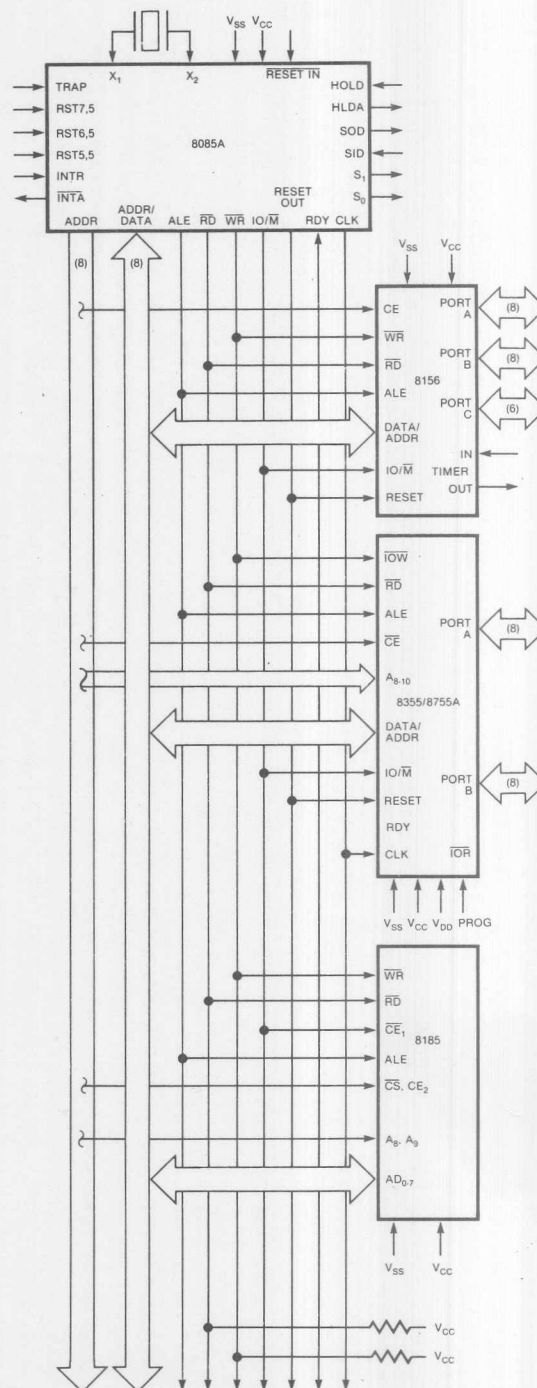


FIGURE 3-9 MCS-85 SYSTEM WITH 8185





**Chapter 4**  
**8080A**  
**Functional**  
**Description**

MOS

8080

MOS

8080



## CHAPTER 4

### THE 8080 CENTRAL PROCESSOR UNIT

The 8080 is a complete 8-bit parallel, central processor unit (CPU) for use in general purpose digital computer systems. It is fabricated on a single LSI chip (see Figure 1-1), using Intel's n-channel silicon gate MOS process. The 8080 transfers data and internal state information via an 8-bit, bidirectional 3-state Data Bus (D<sub>0</sub>-D<sub>7</sub>). Memory and peripheral device addresses are transmitted over a separate 16-

bit 3-state Address Bus (A<sub>0</sub>-A<sub>15</sub>). Six timing and control outputs (SYNC, DBIN, WAIT, WR, HLDA and INTE) emanate from the 8080, while four control inputs (READY, HOLD, INT and RESET), four power inputs (+12v, +5v, -5v, and GND) and two clock inputs ( $\phi_1$  and  $\phi_2$ ) are accepted by the 8080.

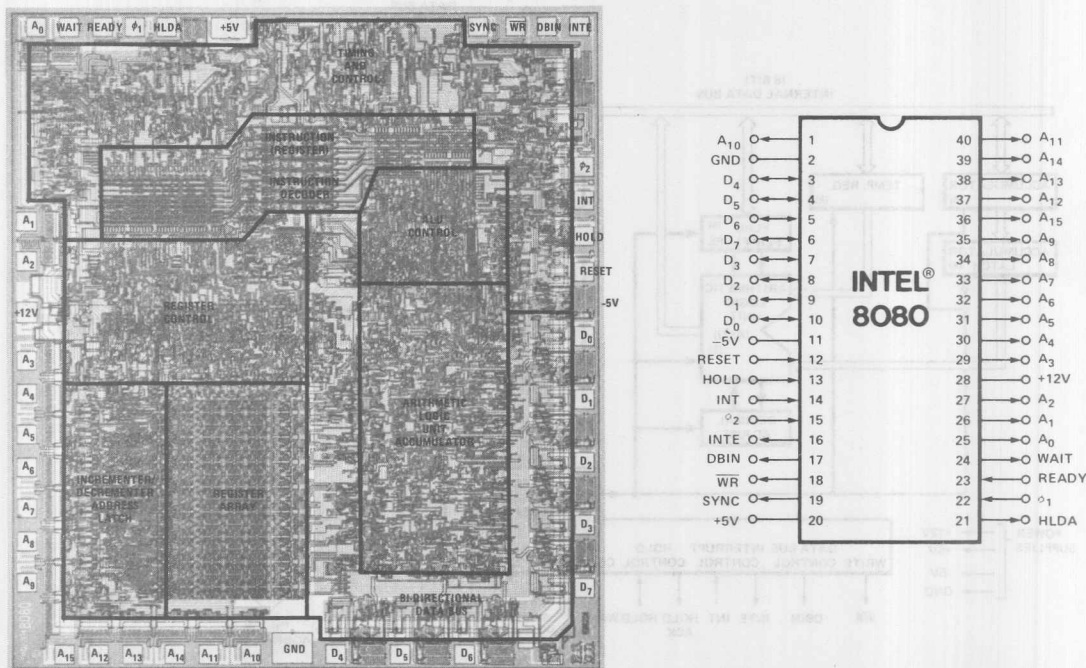


Figure 4-1. 8080 Photomicrograph With Pin Designations



## Arithmetic and Logic Unit (ALU):

The ALU contains the following registers:

- An 8-bit accumulator
- An 8-bit temporary accumulator (ACT)
- A 5-bit flag register: zero, carry, sign, parity and auxiliary carry
- An 8-bit temporary register (TMP)

Arithmetic, logical and rotate operations are performed in the ALU. The ALU is fed by the temporary register (TMP) and the temporary accumulator (ACT) and carry flip-flop. The result of the operation can be transferred to the internal bus or to the accumulator; the ALU also feeds the flag register.

The temporary register (TMP) receives information from the internal bus and can send all or portions of it to the ALU, the flag register and the internal bus.

The accumulator (ACC) can be loaded from the ALU and the internal bus and can transfer data to the temporary accumulator (ACT) and the internal bus. The contents of the accumulator (ACC) and the auxiliary carry flip-flop can be tested for decimal correction during the execution of the DAA instruction (see Section 5).

## Instruction Register and Control:

During an instruction fetch, the first byte of an instruction (containing the OP code) is transferred from the internal bus to the 8-bit instruction register.

The contents of the instruction register are, in turn, available to the instruction decoder. The output of the decoder, combined with various timing signals, provides the control signals for the register array, ALU and data buffer blocks. In addition, the outputs from the instruction decoder and external control signals feed the timing and state control section which generates the state and cycle timing signals.

## Data Bus Buffer:

This 8-bit bidirectional 3-state buffer is used to isolate the CPU's internal bus from the external data bus (D<sub>0</sub> through D<sub>7</sub>). In the output mode, the internal bus content is loaded into an 8-bit latch that, in turn, drives the data bus output buffers. The output buffers are switched off during input or non-transfer operations.

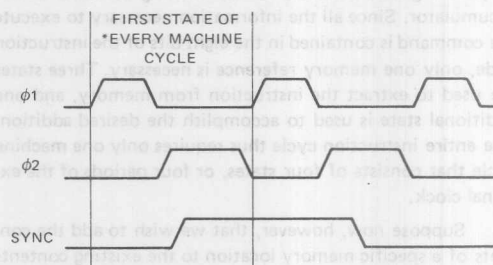
During the input mode, data from the external data bus is transferred to the internal bus. The internal bus is pre-charged at the beginning of each internal state, except for the transfer state (TW and T<sub>3</sub>—described later in this chapter).

## THE PROCESSOR CYCLE

An **instruction cycle** is defined as the time required to fetch and execute an instruction. During the fetch, a selected instruction (one, two or three bytes) is extracted from memory and deposited in the CPU's instruction register. During the execution phase, the instruction is decoded and translated into specific processing activities.

Every instruction cycle consists of one, two, three, four or five machine cycles. A **machine cycle** is required each time the CPU accesses memory or an I/O port. The fetch portion of an instruction cycle requires one machine cycle for each byte to be fetched. The duration of the execution portion of the instruction cycle depends on the kind of instruction that has been fetched. Some instructions do not require any machine cycles other than those necessary to fetch the instruction; other instructions, however, require additional machine cycles to write or read data to/from memory or I/O devices. The DAD instruction is an exception in that it requires two additional machine cycles to complete an internal register-pair add (see Section 5).

Each machine cycle consists of three, four or five states. A state is the smallest unit of processing activity and is defined as the interval between two successive positive-going transitions of the  $\phi_1$  driven clock pulse. The 8080 is driven by a two-phase clock oscillator. All processing activities are referred to the period of this clock. The two non-overlapping clock pulses, labeled  $\phi_1$  and  $\phi_2$ , are furnished by external circuitry. It is the  $\phi_1$  clock pulse which divides each machine cycle into states. Timing logic within the 8080 uses the clock inputs to produce a SYNC pulse, which identifies the beginning of every machine cycle. The SYNC pulse is triggered by the low-to-high transition of  $\phi_2$ , as shown in Figure 4-3.



\*SYNC DOES NOT OCCUR IN THE SECOND AND THIRD MACHINE CYCLES OF A DAD INSTRUCTION SINCE THESE MACHINE CYCLES ARE USED FOR AN INTERNAL REGISTER-PAIR ADD.

Figure 4-3.  $\phi_1$ ,  $\phi_2$  and SYNC Timing

There are three exceptions to the defined duration of a state. They are the WAIT state, the hold (HLDA) state and the halt (HLTA) state, described later in this chapter. Because the WAIT, the HLDA, and the HLTA states depend upon external events, they are by their nature of indeterminate length. Even these exceptional states, however, must



be synchronized with the pulses of the driving clock. Thus, the duration of all states are integral multiples of the clock period.

To summarize then, each clock period marks a state; three to five states constitute a machine cycle; and one to five machine cycles comprise an instruction cycle. A full instruction cycle requires anywhere from four to eight-teen states for its completion, depending on the kind of instruction involved.

### Machine Cycle Identification:

With the exception of the DAD instruction, there is just one consideration that determines how many machine cycles are required in any given instruction cycle: the number of times that the processor must reference a memory address or an addressable peripheral device, in order to fetch and execute the instruction. Like many processors, the 8080 is so constructed that it can transmit only one address per machine cycle. Thus, if the fetch and execution of an instruction requires two memory references, then the instruction cycle associated with that instruction consists of two machine cycles. If five such references are called for, then the instruction cycle contains five machine cycles.

Every instruction cycle has at least one reference to memory, during which the instruction is fetched. An instruction cycle must always have a fetch, even if the execution of the instruction requires no further references to memory. The first machine cycle in every instruction cycle is therefore a FETCH. Beyond that, there are no fast rules. It depends on the kind of instruction that is fetched.

Consider some examples. The add-register (ADD r) instruction is an instruction that requires only a single machine cycle (FETCH) for its completion. In this one-byte instruction, the contents of one of the CPU's six general purpose registers is added to the existing contents of the accumulator. Since all the information necessary to execute the command is contained in the eight bits of the instruction code, only one memory reference is necessary. Three states are used to extract the instruction from memory, and one additional state is used to accomplish the desired addition. The entire instruction cycle thus requires only one machine cycle that consists of four states, or four periods of the external clock.

Suppose now, however, that we wish to add the contents of a specific memory location to the existing contents of the accumulator (ADD M). Although this is quite similar in principle to the example just cited, several additional steps will be used. An extra machine cycle will be used, in order to address the desired memory location.

The actual sequence is as follows. First the processor extracts from memory the one-byte instruction word addressed by its program counter. This takes three states. The eight-bit instruction word obtained during the FETCH machine cycle is deposited in the CPU's instruction register and used to direct activities during the remainder of the instruction cycle. Next, the processor sends out, as an address,

the contents of its H and L registers. The eight-bit data word returned during this MEMORY READ machine cycle is placed in a temporary register inside the 8080 CPU. By now three more clock periods (states) have elapsed. In the seventh and final state, the contents of the temporary register are added to those of the accumulator. Two machine cycles, consisting of seven states in all, complete the "ADD M" instruction cycle.

At the opposite extreme is the save H and L registers (SHLD) instruction, which requires five machine cycles. During an "SHLD" instruction cycle, the contents of the processor's H and L registers are deposited in two sequentially adjacent memory locations; the destination is indicated by two address bytes which are stored in the two memory locations immediately following the operation code byte. The following sequence of events occurs:

- (1) A FETCH machine cycle, consisting of four states. During the first three states of this machine cycle, the processor fetches the instruction indicated by its program counter. The program counter is then incremented. The fourth state is used for internal instruction decoding.
- (2) A MEMORY READ machine cycle, consisting of three states. During this machine cycle, the byte indicated by the program counter is read from memory and placed in the processor's Z register. The program counter is incremented again.
- (3) Another MEMORY READ machine cycle, consisting of three states, in which the byte indicated by the processor's program counter is read from memory and placed in the W register. The program counter is incremented, in anticipation of the next instruction fetch.
- (4) A MEMORY WRITE machine cycle, of three states, in which the contents of the L register are transferred to the memory location pointed to by the present contents of the W and Z registers. The state following the transfer is used to increment the W,Z register pair so that it indicates the next memory location to receive data.
- (5) A MEMORY WRITE machine cycle, of three states, in which the contents of the H register are transferred to the new memory location pointed to by the W,Z register pair.

In summary, the "SHLD" instruction cycle contains five machine cycles and takes 16 states to execute.

Most instructions fall somewhere between the extremes typified by the "ADD r" and the "SHLD" instructions. The input (IN) and the output (OUT) instructions, for example, require three machine cycles: a FETCH, to obtain the instruction; a MEMORY READ, to obtain the address of the object peripheral; and an INPUT or an OUTPUT machine cycle, to complete the transfer.

While no one instruction cycle will consist of more than five machine cycles, the following ten different types of machine cycles may occur within an instruction cycle:

- (1) FETCH (M1)
- (2) MEMORY READ
- (3) MEMORY WRITE
- (4) STACK READ
- (5) STACK WRITE
- (6) INPUT
- (7) OUTPUT
- (8) INTERRUPT
- (9) HALT
- (10) HALT • INTERRUPT

The machine cycles that actually do occur in a particular instruction cycle depend upon the kind of instruction, with the overriding stipulation that the first machine cycle in any instruction cycle is always a FETCH.

The processor identifies the machine cycle in progress by transmitting an eight-bit status word during the first state of every machine cycle. Updated status information is presented on the 8080's data lines (D<sub>0</sub>-D<sub>7</sub>), during the SYNC interval. This data should be saved in latches, and used to develop control signals for external circuitry. Table 4-1 shows how the positive-true status information is distributed on the processor's data bus.

Status signals are provided principally for the control of external circuitry. Simplicity of interface, rather than machine cycle identification, dictates the logical definition of individual status bits. You will therefore observe that certain processor machine cycles are uniquely identified by a single status bit, but that others are not. The M<sub>1</sub> status bit (D<sub>5</sub>), for example, unambiguously identifies a FETCH machine cycle. A STACK READ, on the other hand, is indicated by the coincidence of STACK and MEMR signals. Machine cycle identification data is also valuable in the test and de-bugging phases of system development. Table 4-1 lists the status bit outputs for each type of machine cycle.

### State Transition Sequence:

Every machine cycle within an instruction cycle consists of three to six active states (referred to as T<sub>1</sub>, T<sub>2</sub>, T<sub>3</sub>, T<sub>4</sub>, T<sub>5</sub> or T<sub>W</sub>). The actual number of states depends upon the instruction being executed, and on the particular machine cycle within the greater instruction cycle. The state transition diagram in Figure 4-4 shows how the 8080 proceeds from state to state in the course of a machine cycle. The diagram also shows how the READY, HOLD, and INTERRUPT lines are sampled during the machine cycle, and how the conditions on these lines may modify the

basic transition sequence. In the present discussion, we are concerned only with the basic sequence and with the READY function. The HOLD and INTERRUPT functions will be discussed later.

The 8080 CPU does not directly indicate its internal state by transmitting a "state control" output during each state; instead, the 8080 supplies direct control output (INTE, HLDA, DBIN, WR and WAIT) for use by external circuitry.

Recall that the 8080 passes through at least three states in every machine cycle, with each state defined by successive low-to-high transitions of the  $\phi_1$  clock. Figure 4-5 shows the timing relationships in a typical FETCH machine cycle. Events that occur in each state are referenced to transitions of the  $\phi_1$  and  $\phi_2$  clock pulses.

The SYNC signal identifies the first state (T<sub>1</sub>) in every machine cycle. As shown in Figure 4-5, the SYNC signal is related to the leading edge of the  $\phi_2$  clock. There is a delay ( $t_{DC}$ ) between the low-to-high transition of  $\phi_2$  and the positive-going edge of the SYNC pulse. There also is a corresponding delay (also  $t_{DC}$ ) between the next  $\phi_2$  pulse and the falling edge of the SYNC signal. Status information is displayed on D<sub>0</sub>-D<sub>7</sub> during the same  $\phi_2$  to  $\phi_2$  interval. Switching of the status signals is likewise controlled by  $\phi_2$ .

The rising edge of  $\phi_2$  during T<sub>1</sub> also loads the processor's address lines (A<sub>0</sub>-A<sub>15</sub>). These lines become stable within a brief delay ( $t_{DA}$ ) of the  $\phi_2$  clocking pulse, and they remain stable until the first  $\phi_2$  pulse after state T<sub>3</sub>. This gives the processor ample time to read the data returned from memory.

Once the processor has sent an address to memory, there is an opportunity for the memory to request a WAIT. This it does by pulling the processor's READY line low, prior to the "Ready set-up" interval ( $t_{RS}$ ) which occurs during the  $\phi_2$  pulse within state T<sub>2</sub> or T<sub>W</sub>. As long as the READY line remains low, the processor will idle, giving the memory time to respond to the addressed data request. Refer to Figure 4-5.

The processor responds to a wait request by entering an alternative state (T<sub>W</sub>) at the end of T<sub>2</sub>, rather than proceeding directly to the T<sub>3</sub> state. Entry into the T<sub>W</sub> state is indicated by a WAIT signal from the processor, acknowledging the memory's request. A low-to-high transition on the WAIT line is triggered by the rising edge of the  $\phi_1$  clock and occurs within a brief delay ( $t_{DC}$ ) of the actual entry into the T<sub>W</sub> state.

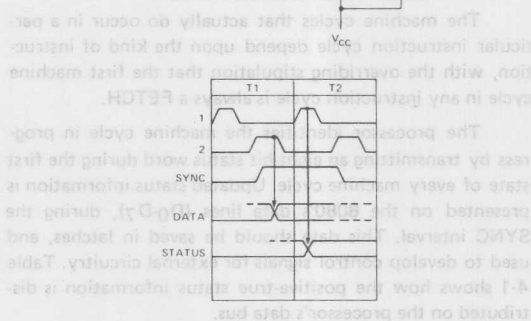
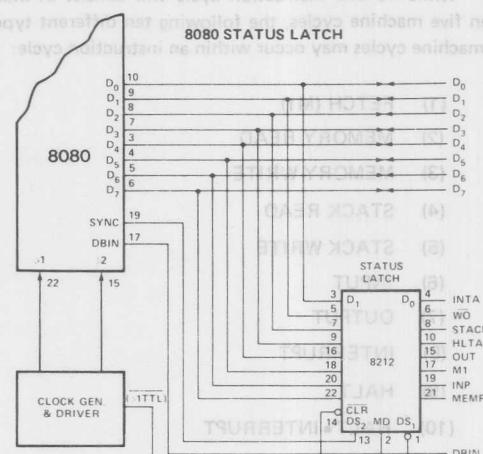
A wait period may be of indefinite duration. The processor remains in the waiting condition until its READY line again goes high. A READY indication must precede the falling edge of the  $\phi_2$  clock by a specified interval ( $t_{RS}$ ), in order to guarantee an exit from the T<sub>W</sub> state. The cycle may then proceed, beginning with the rising edge of the next  $\phi_1$  clock. A WAIT interval will therefore consist of an integral number of T<sub>W</sub> states and will always be a multiple of the clock period.

Instructions for the 8080 require from one to five machine cycles for complete execution. The 8080 sends out 8 bit of status information on the data bus at the beginning of each machine cycle (during SYNC time). The following table defines the status information.

#### STATUS INFORMATION DEFINITION

Symbols	Bit	Definition
INTA*	D <sub>0</sub>	Acknowledge signal for INTERRUPT request. Signal should be used to gate a re-start instruction onto the data bus when DBIN is active.
W $\overline{O}$	D <sub>1</sub>	Indicates that the operation in the current machine cycle will be a WRITE memory or OUTPUT function (W $\overline{O}$ = 0). Otherwise, a READ memory or INPUT operation will be executed.
STACK	D <sub>2</sub>	Indicates that the address bus holds the pushdown stack address from the Stack Pointer.
HLTA	D <sub>3</sub>	Acknowledge signal for HALT instruction.
OUT	D <sub>4</sub>	Indicates that the address bus contains the address of an output device and the data bus will contain the output data when W $\overline{R}$ is active.
M <sub>1</sub>	D <sub>5</sub>	Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instruction.
INP*	D <sub>6</sub>	Indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when DBIN is active.
MEMR*	D <sub>7</sub>	Designates that the data bus will be used for memory read data.

\*These three status bits can be used to control the flow of data onto the 8080 data bus.



#### STATUS WORD CHART

		TYPE OF MACHINE CYCLE											
		DATA BUS BIT	STATUS INFORMATION	INSTRUCTION FETCH	MEMORY READ	MEMORY WRITE	STACK READ	STACK WRITE	INPUT READ	OUTPUT WRITE	INTERRUPT ACKNOWLEDGE	HALT ACKNOWLEDGE	INTERRUPT ACKNOWLEDGE WHILE HALT
			①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	↖ (N) STATUS WORD
D <sub>0</sub>	INTA	0	0	0	0	0	0	0	0	1	0	1	
D <sub>1</sub>	W $\overline{O}$	1	1	0	1	0	1	0	1	1	1	1	
D <sub>2</sub>	STACK	0	0	0	1	1	0	0	0	0	0	0	
D <sub>3</sub>	HLTA	0	0	0	0	0	0	0	0	0	1	1	
D <sub>4</sub>	OUT	0	0	0	0	0	0	0	1	0	0	0	
D <sub>5</sub>	M <sub>1</sub>	1	0	0	0	0	0	0	0	1	0	1	
D <sub>6</sub>	INP	0	0	0	0	0	0	1	0	0	0	0	
D <sub>7</sub>	MEMR	1	1	0	1	0	0	0	0	0	1	0	

Table 4-1. 8080 Status Bit Definitions



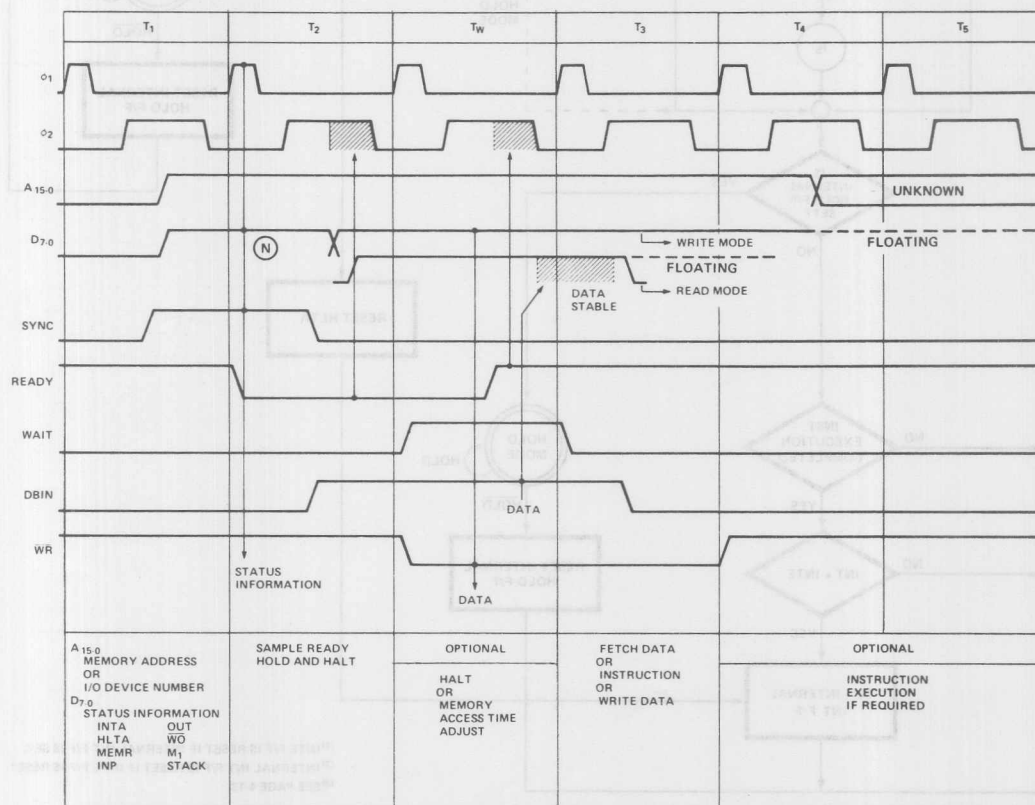
During a machine cycle, the processor interprets the data on its data bus as an instruction. During a MEMORY READ or a STACK READ, data on this bus is interpreted as a data word. The processor outputs data on this bus during a MEMORY WRITE machine cycle. During I/O operations, the processor may either transmit or receive data, depending on whether an OUTPUT or an INPUT operation is involved.

Figure 4-7 illustrates the timing that is characteristic of a data input operation. As shown, the low-to-high transition of  $\phi_2$  during  $T_2$  clears status information from the processor's data lines, preparing these lines for the receipt of incoming data. The data presented to the processor must have stabilized prior to both the " $\phi_1$ -data set-up" interval ( $t_{DS1}$ ), that precedes the falling edge of the  $\phi_1$  pulse defining state  $T_3$ , and the " $\phi_2$ -data set-up" interval ( $t_{DS2}$ ), that precedes the rising edge of  $\phi_2$  in state  $T_3$ . This same

Data placed on these lines by memory or by other external devices will be sampled during  $T_3$ .

During the input of data to the processor, the 8080 generates a DBIN signal which should be used externally to enable the transfer. Machine cycles in which DBIN is available include: FETCH, MEMORY READ, STACK READ, and INTERRUPT. DBIN is initiated by the rising edge of  $\phi_2$  during state  $T_2$  and terminated by the corresponding edge of  $\phi_2$  during  $T_3$ . Any  $T_W$  phases intervening between  $T_2$  and  $T_3$  will therefore extend DBIN by one or more clock periods.

Figure 4-7 shows the timing of a machine cycle in which the processor outputs data. Output data may be destined either for memory or for peripherals. The rising edge of  $\phi_2$  within state  $T_2$  clears status information from the CPU's data lines, and loads in the data which is to be output to external devices. This substitution takes place within the



NOTE: (N) Refer to Status Word Chart on Page 4-6.

Figure 4-5. Basic 8080 Instruction Cycle



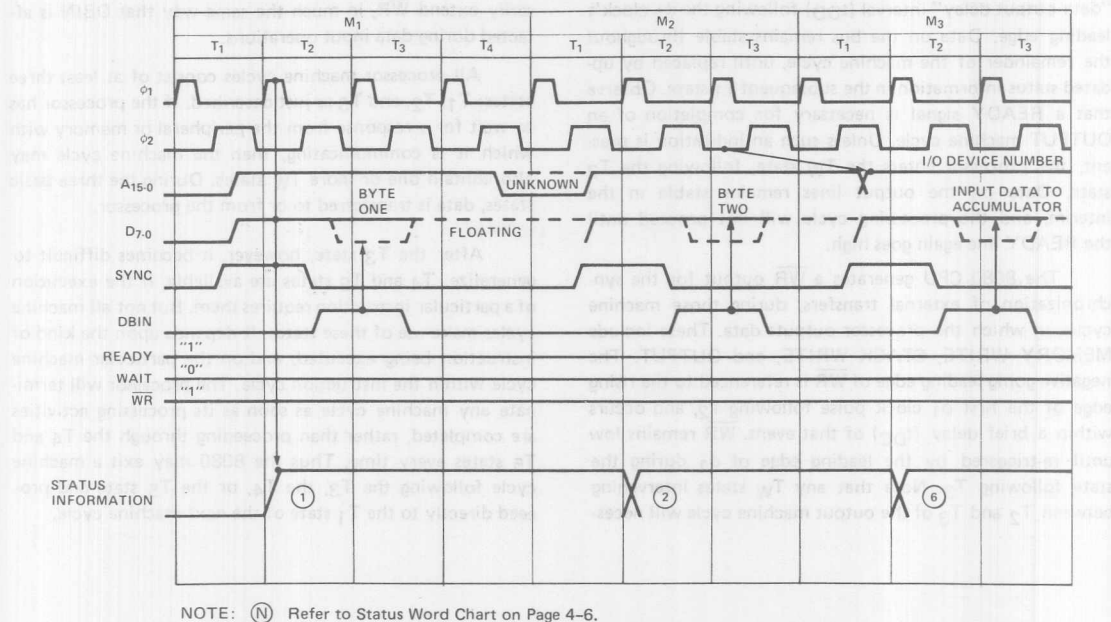


Figure 4-6. Input Instruction Cycle

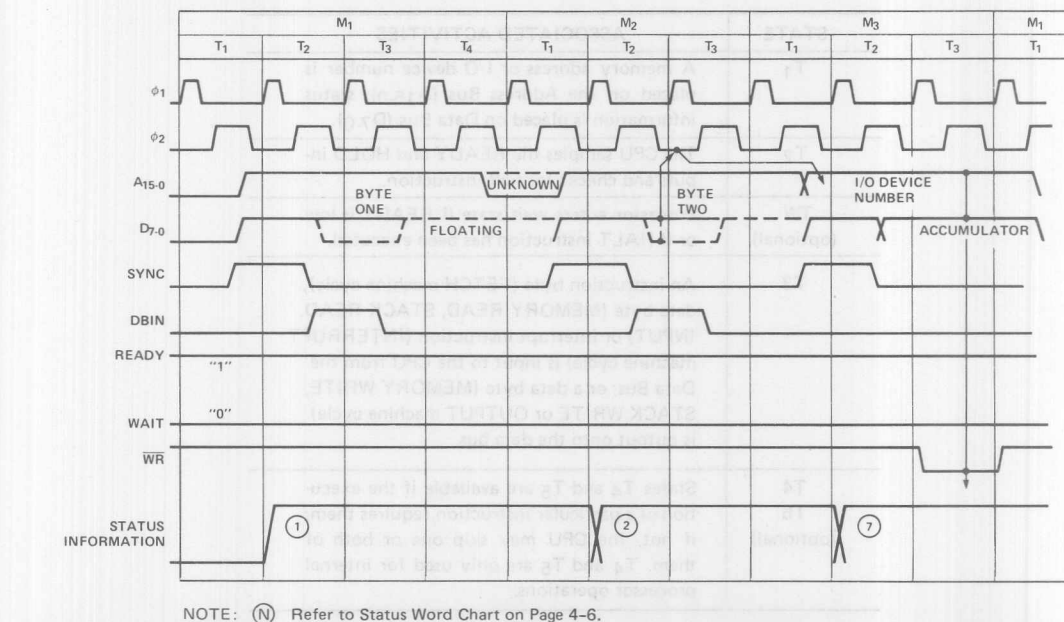


Figure 4-7. Output Instruction Cycle

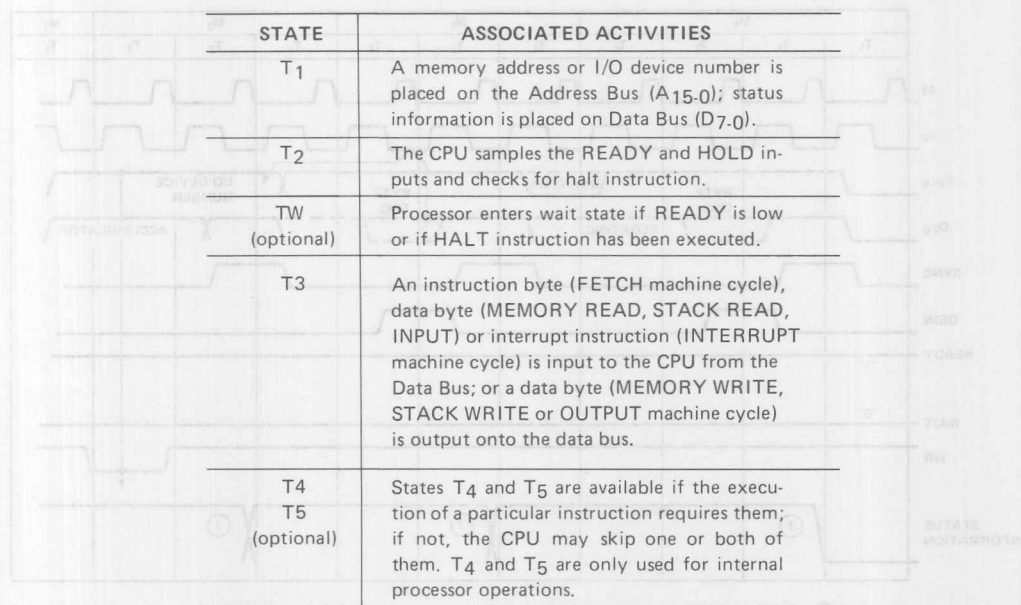
"data output delay" interval ( $t_{DD}$ ) following the  $\phi_2$  clock's leading edge. Data on the bus remains stable throughout the remainder of the machine cycle, until replaced by updated status information in the subsequent  $T_1$  state. Observe that a **READY** signal is necessary for completion of an **OUTPUT** machine cycle. Unless such an indication is present, the processor enters the  $T_W$  state, following the  $T_2$  state. Data on the output lines remains stable in the interim, and the processing cycle will not proceed until the **READY** line again goes high.

The 8080 CPU generates a  $\overline{WR}$  output for the synchronization of external transfers, during those machine cycles in which the processor outputs data. These include **MEMORY WRITE**, **STACK WRITE**, and **OUTPUT**. The negative-going leading edge of  $\overline{WR}$  is referenced to the rising edge of the first  $\phi_1$  clock pulse following  $T_2$ , and occurs within a brief delay ( $t_{DC}$ ) of that event.  $\overline{WR}$  remains low until re-triggered by the leading edge of  $\phi_1$  during the state following  $T_3$ . Note that any  $T_W$  states intervening between  $T_2$  and  $T_3$  of the output machine cycle will neces-

sarily extend  $\overline{WR}$ , in much the same way that **DBIN** is affected during data input operations.

All processor machine cycles consist of at least three states:  $T_1$ ,  $T_2$ , and  $T_3$  as just described. If the processor has to wait for a response from the peripheral or memory with which it is communicating, then the machine cycle may also contain one or more  $T_W$  states. During the three basic states, data is transferred to or from the processor.

After the  $T_3$  state, however, it becomes difficult to generalize.  $T_4$  and  $T_5$  states are available, if the execution of a particular instruction requires them. But not all machine cycles make use of these states. It depends upon the kind of instruction being executed, and on the particular machine cycle within the instruction cycle. The processor will terminate any machine cycle as soon as its processing activities are completed, rather than proceeding through the  $T_4$  and  $T_5$  states every time. Thus the 8080 may exit a machine cycle following the  $T_3$ , the  $T_4$ , or the  $T_5$  state and proceed directly to the  $T_1$  state of the next machine cycle.



STATE	ASSOCIATED ACTIVITIES
$T_1$	A memory address or I/O device number is placed on the Address Bus ( $A_{15:0}$ ); status information is placed on Data Bus ( $D_{7:0}$ ).
$T_2$	The CPU samples the <b>READY</b> and <b>HOLD</b> inputs and checks for halt instruction.
$T_W$ (optional)	Processor enters wait state if <b>READY</b> is low or if <b>HALT</b> instruction has been executed.
$T_3$	An instruction byte ( <b>FETCH</b> machine cycle), data byte ( <b>MEMORY READ</b> , <b>STACK READ</b> , <b>INPUT</b> ) or interrupt instruction ( <b>INTERRUPT</b> machine cycle) is input to the CPU from the Data Bus; or a data byte ( <b>MEMORY WRITE</b> , <b>STACK WRITE</b> or <b>OUTPUT</b> machine cycle) is output onto the data bus.
$T_4$ $T_5$ (optional)	States $T_4$ and $T_5$ are available if the execution of a particular instruction requires them; if not, the CPU may skip one or both of them. $T_4$ and $T_5$ are only used for internal processor operations.

Table 4-2. State Definitions

## INTERRUPT SEQUENCES

The 8080 has the built-in capacity to handle external interrupt requests. A peripheral device can initiate an interrupt simply by driving the processor's interrupt (INT) line high.

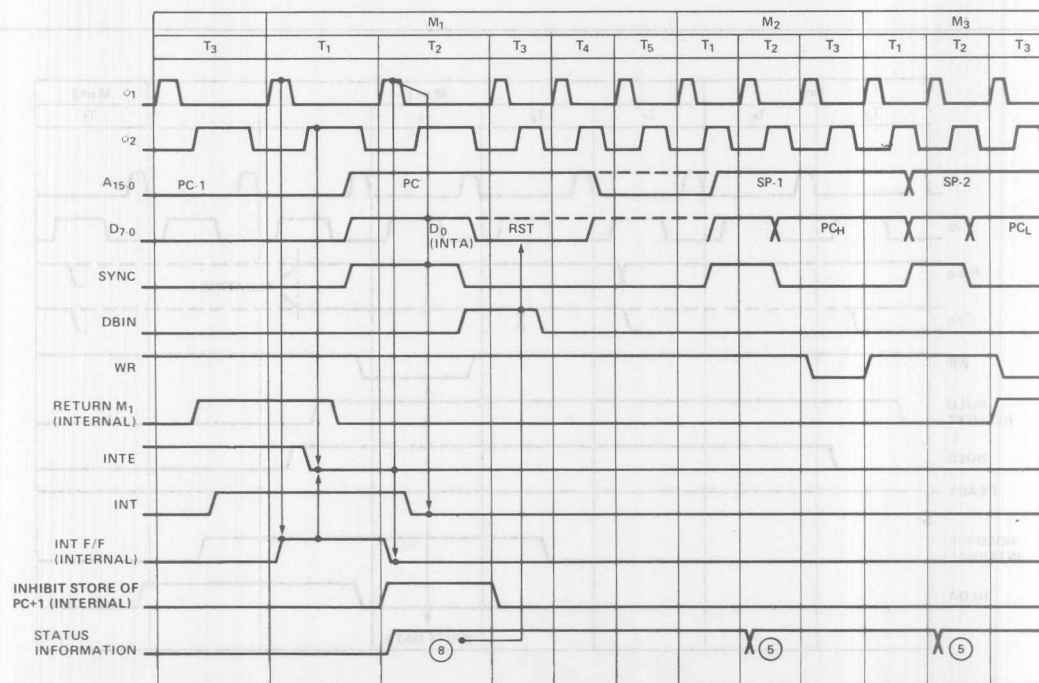
The interrupt (INT) input is asynchronous, and a request may therefore originate at any time during any instruction cycle. Internal logic re-clocks the external request, so that a proper correspondence with the driving clock is established. As Figure 4-8 shows, an interrupt request (INT) arriving during the time that the interrupt enable line (INTE) is high, acts in coincidence with the  $\phi_2$  clock to set the internal interrupt latch. This event takes place during the last state of the instruction cycle in which the request occurs, thus ensuring that any instruction in progress is completed before the interrupt can be processed.

The INTERRUPT machine cycle which follows the arrival of an enabled interrupt request resembles an ordinary FETCH machine cycle in most respects. The  $M_1$  status bit is transmitted as usual during the SYNC interval. It is accompanied, however, by an INTA status bit ( $D_0$ ) which acknowledges the external request. The contents of the program counter are latched onto the CPU's address lines during  $T_1$ , but the counter itself is not incremented during the INTERRUPT machine cycle, as it otherwise would be.

In this way, the pre-interrupt status of the program counter is preserved, so that data in the counter may be restored by the interrupted program after the interrupt request has been processed.

The interrupt cycle is otherwise indistinguishable from an ordinary FETCH machine cycle. The processor itself takes no further special action. It is the responsibility of the peripheral logic to see that an eight-bit interrupt instruction is "jammed" onto the processor's data bus during state  $T_3$ . In a typical system, this means that the data-in bus from memory must be temporarily disconnected from the processor's main data bus, so that the interrupting device can command the main bus without interference.

The 8080's instruction set provides a special one-byte call which facilitates the processing of interrupts (the ordinary program Call takes three bytes). This is the RESTART instruction (RST). A variable three-bit field embedded in the eight-bit field of the RST enables the interrupting device to direct a Call to one of eight fixed memory locations. The decimal addresses of these dedicated locations are: 0, 8, 16, 24, 32, 40, 48, and 56. Any of these addresses may be used to store the first instruction(s) of a routine designed to service the requirements of an interrupting device. Since the (RST) is a call, completion of the instruction also stores the old program counter contents on the STACK.



NOTE: (N) Refer to Status Word Chart on Page 4-6.

Figure 4-8. Interrupt Timing

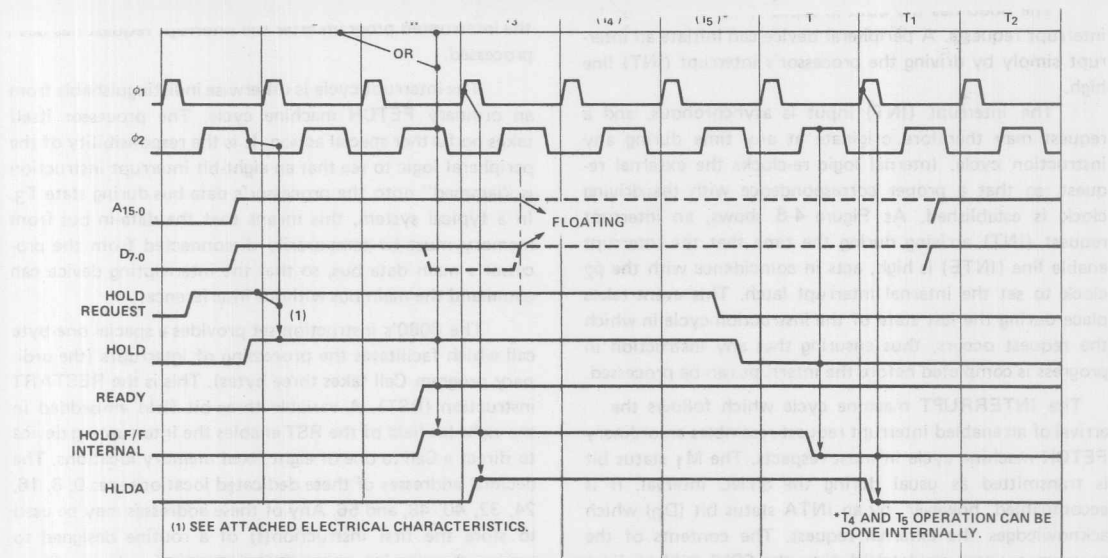


Figure 4-9. HOLD Operation (Read Mode)

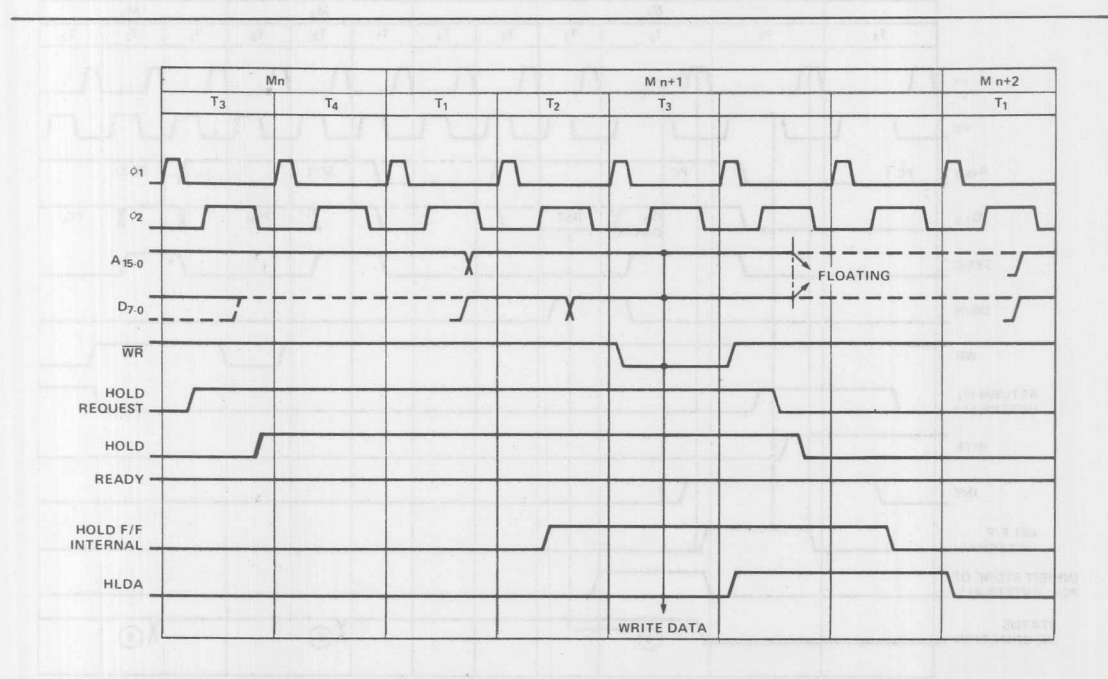


Figure 4-10. HOLD Operation (Write Mode)

## HOLD SEQUENCES

The 8080A CPU contains provisions for Direct Memory Access (DMA) operations. By applying a HOLD to the appropriate control pin on the processor, an external device can cause the CPU to suspend its normal operations and relinquish control of the address and data busses. The processor responds to a request of this kind by floating its address to other devices sharing the busses. At the same time, the processor acknowledges the HOLD by placing a high on its HLDA output pin. During an acknowledged HOLD, the address and data busses are under control of the peripheral which originated the request, enabling it to conduct memory transfers without processor intervention.

Like the interrupt, the HOLD input is synchronized internally. A HOLD signal must be stable prior to the "Hold set-up" interval ( $t_{HS}$ ), that precedes the rising edge of  $\phi_2$ .

Figures 4-9 and 4-10 illustrate the timing involved in HOLD operations. Note the delay between the asynchronous HOLD REQUEST and the re-clocked HOLD. As shown in the diagram, a coincidence of the READY, the HOLD, and the  $\phi_2$  clocks sets the internal hold latch. Setting the latch enables the subsequent rising edge of the  $\phi_1$  clock pulse to trigger the HLDA output as described below.

Acknowledgement of the HOLD REQUEST precedes slightly the actual floating of the processor's address and data lines. The processor acknowledges a HOLD at the beginning of T<sub>3</sub>, if a read or an input machine cycle is in progress (see Figure 4-9). Otherwise, acknowledgement is deferred until the beginning of the state following T<sub>3</sub> (see Figure 4-10). In both cases, however, the HLDA goes high within a specified delay ( $t_{DC}$ ) of the rising edge of the selected  $\phi_1$  clock pulse. Address and data lines are floated within a brief delay after the rising edge of the next  $\phi_2$  clock pulse. This relationship is also shown in the diagrams.

To all outward appearances, the processor has suspended its operations once the address and data busses are floated. Internally, however, certain functions may continue. If a HOLD REQUEST is acknowledged at T<sub>3</sub>, and if the processor is in the middle of a machine cycle which requires four or more states to complete, the CPU proceeds through T<sub>4</sub> and T<sub>5</sub> before coming to a rest. Not until the end of the machine cycle is reached will processing activities cease. Internal processing is thus permitted to overlap the external DMA transfer, improving both the efficiency and the speed of the entire system.

The processor exits the holding state through a sequence similar to that by which it entered. A HOLD REQUEST is terminated asynchronously when the external device has completed its data transfer. The HLDA output

returns to a low level following the leading edge of the next  $\phi_1$  clock pulse. Normal processing resumes with the machine cycle following the last cycle that was executed.

## HALT SEQUENCES

When a halt instruction (HLT) is executed, the CPU enters the halt state (T<sub>WH</sub>) after state T<sub>2</sub> of the next machine cycle, as shown in Figure 4-11. There are only three ways in which the 8080 can exit the halt state:

- A high on the RESET line will always reset the 8080 to state T<sub>1</sub>; RESET also clears the program counter.
- A HOLD input will cause the 8080 to enter the hold state, as previously described. When the HOLD line goes low, the 8080 re-enters the halt state on the rising edge of the next  $\phi_1$  clock pulse.
- An interrupt (i.e., INT goes high while INTE is enabled) will cause the 8080 to exit the Halt state and enter state T<sub>1</sub> on the rising edge of the next  $\phi_1$  clock pulse. NOTE: The interrupt enable (INTE) flag **must** be set when the halt state is entered; otherwise, the 8080 will only be able to exit via a RESET signal.

Figure 4-12 illustrates halt sequencing in flow chart form.

## START-UP OF THE 8080 CPU

When power is applied initially to the 8080, the processor begins operating immediately. The contents of its program counter, stack pointer, and the other working registers are naturally subject to random factors and cannot be specified. For this reason, it will be necessary to begin the power-up sequence with RESET.

An external RESET signal of three clock period duration (minimum) restores the processor's internal program counter to zero. Program execution thus begins with memory location zero, following a RESET. Systems which require the processor to wait for an explicit start-up signal will store a halt instruction (EI, HLT) in the first two locations. A manual or an automatic INTERRUPT will be used for starting. In other systems, the processor may begin executing its stored program immediately. Note, however, that the RESET has no effect on status flags, or on any of the processor's working registers (accumulator, registers, or stack pointer). The contents of these registers remain indeterminate, until initialized explicitly by the program.

Figure 4-12. HALT Sequencing Flow Chart



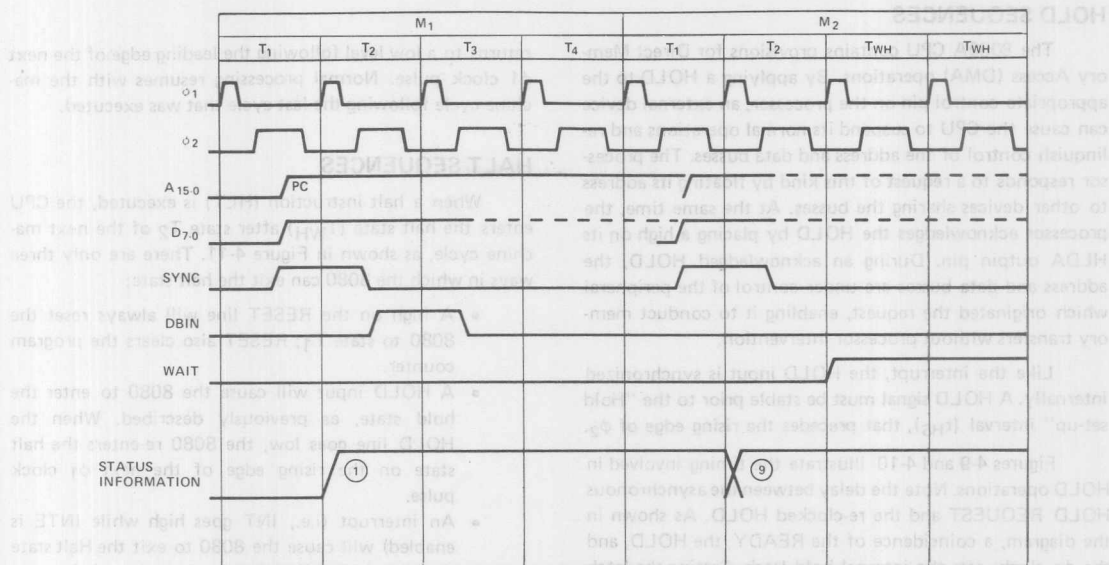


Figure 4-11. HALT Timing

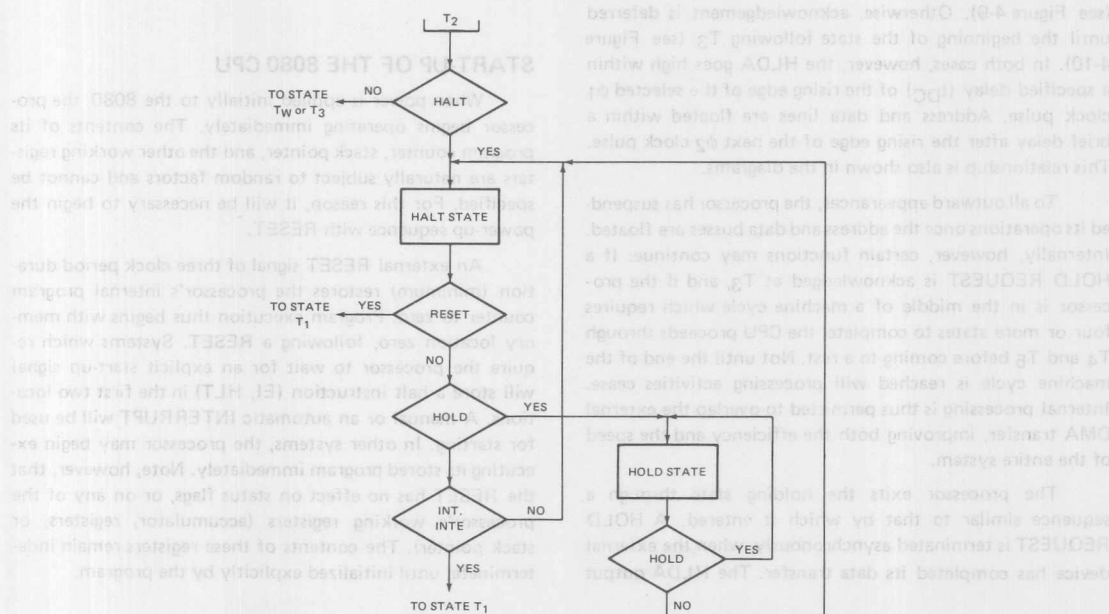
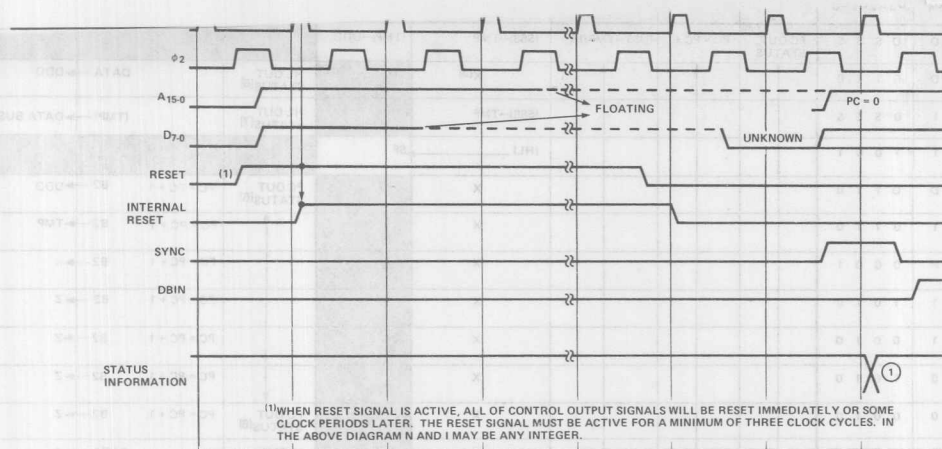


Figure 4-12. HALT Sequence Flow Chart



NOTE: (N) Refer to Status Word Chart on Page 4-6.

Figure 4-13. Reset

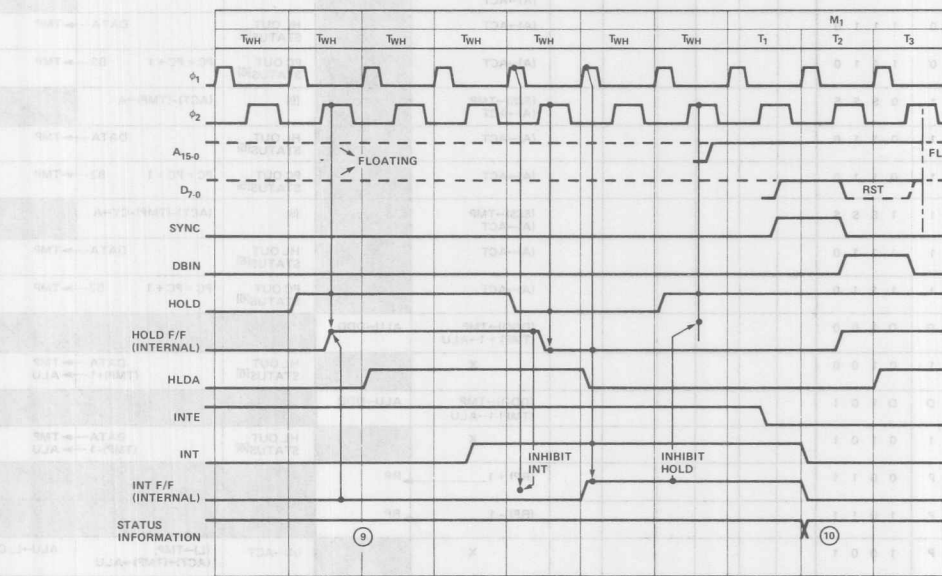


Figure 4-14. Relation between HOLD and INT in the HALT State

MNEMONIC	OP CODE								M1[1]					M2		
	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	T1	T <sub>2</sub> [2]	T3	T4	T5	T1	T <sub>2</sub> [2]	T3
MOV r <sub>1</sub> , r <sub>2</sub>	0	1	D	D	D	S	S	S	PC OUT STATUS	PC = PC + 1	INST-TMP/IR	(SSS)-TMP	(TMP)-DDD			
MOV r, M	0	1	D	D	D	1	1	0				X[3]		HL OUT STATUS[6]	DATA	→DDD
MOV M, r	0	1	1	1	0	S	S	S				(SSS)-TMP		HL OUT STATUS[7]	(TMP)	→DATA BUS
SPHL	1	1	1	1	1	0	0	1				(HL) → SP				
MVI r, data	0	0	D	D	D	1	1	0				X		PC OUT STATUS[6]	PC = PC + 1 B <sub>2</sub>	→DDD
MVI M, data	0	0	1	1	0	1	1	0				X			PC = PC + 1 B <sub>2</sub>	→TMP
LXI rp, data	0	0	R	P	0	0	0	1				X			PC = PC + 1 B <sub>2</sub>	→r <sub>1</sub>
LDA addr	0	0	1	1	1	0	1	0				X			PC = PC + 1 B <sub>2</sub>	→Z
STA addr	0	0	1	1	0	0	1	0				X			PC = PC + 1 B <sub>2</sub>	→Z
LHLD addr	0	0	1	0	1	0	1	0				X			PC = PC + 1 B <sub>2</sub>	→Z
SHLD addr	0	0	1	0	0	0	1	0				X		PC OUT STATUS[6]	PC = PC + 1 B <sub>2</sub>	→Z
LDAX rp[4]	0	0	R	P	1	0	1	0				X		rp OUT STATUS[6]	DATA	→A
STAX rp[4]	0	0	R	P	0	0	1	0				X		rp OUT STATUS[7]	(A)	→DATA BUS
XCHG	1	1	1	0	1	0	1	1				X		[11]	(HL) ... (DE)	
ADD r	1	0	0	0	0	S	S	S				(SSS)-TMP (A)-ACT		[9]	(ACT)+(TMP)-A	
ADD M	1	0	0	0	0	1	1	0				(A)-ACT		HL OUT STATUS[6]	DATA	→TMP
ADI data	1	1	0	0	0	1	1	0				(A)-ACT		PC OUT STATUS[6]	PC = PC + 1 B <sub>2</sub>	→TMP
ADC r	1	0	0	0	1	S	S	S				(SSS)-TMP (A)-ACT		[9]	(ACT)+(TMP)+CY-A	
ADC M	1	0	0	0	1	1	1	0				(A)-ACT		HL OUT STATUS[6]	DATA	→TMP
ACI data	1	1	0	0	1	1	1	0				(A)-ACT		PC OUT STATUS[6]	PC = PC + 1 B <sub>2</sub>	→TMP
SUB r	1	0	0	1	0	S	S	S				(SSS)-TMP (A)-ACT		[9]	(ACT)-(TMP)-A	
SUB M	1	0	0	1	0	1	1	0				(A)-ACT		HL OUT STATUS[6]	DATA	→TMP
SUI data	1	1	0	1	0	1	1	0				(A)-ACT		PC OUT STATUS[6]	PC = PC + 1 B <sub>2</sub>	→TMP
SBB r	1	0	0	1	1	S	S	S				(SSS)-TMP (A)-ACT		[9]	(ACT)-(TMP)-CY-A	
SBB M	1	0	0	1	1	1	1	0				(A)-ACT		HL OUT STATUS[6]	DATA	→TMP
SBI data	1	1	0	1	1	1	1	0				(A)-ACT		PC OUT STATUS[6]	PC = PC + 1 B <sub>2</sub>	→TMP
INR r	0	0	D	D	D	1	0	0				(DDD)-TMP (TMP) + 1-ALU	ALU-DDD			
INR M	0	0	1	1	0	1	0	0				X		HL OUT STATUS[6]	DATA (TMP)+1	→TMP →ALU
DCR r	0	0	D	D	D	1	0	1				(DDD)-TMP (TMP)-1-ALU	ALU-DDD			
DCR M	0	0	1	1	0	1	0	1				X		HL OUT STATUS[6]	DATA (TMP)-1	→TMP →ALU
INX rp	0	0	R	P	0	0	1	1				(RP) + 1 → RP				
DCX rp	0	0	R	P	1	0	1	1				(RP) - 1 → RP				
DAD rp[8]	0	0	R	P	1	0	0	1				X		(r <sub>1</sub> )-ACT	(L)-TMP, (ACT)+(TMP)-ALU	ALU-L, CY
DAA	0	0	1	0	0	1	1	1				66-ACT [10] (A)-TMP		[9]	DAA-A FLAGS [10]	
ANA r	1	0	1	0	0	S	S	S				(SSS)-TMP (A)-ACT		[9]	(ACT)+(TMP)-A	
ANA M	1	0	1	0	0	1	1	0	PC OUT STATUS	PC = PC + 1	INST-TMP/IR	(A)-ACT		HL OUT STATUS[6]	DATA	→TMP



MNEMONIC	OP CODE		M1[1]					M2		
			T1	T2[2]	T3	T4	T5	T1	T2[2]	T3
ANI data	1 1 1 0	0 1 1 0	PC OUT STATUS	PC = PC + 1	INST-TMP/IR	(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2	→TMP
XRA r	1 0 1 0	1 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)+(TPM)→A	
XRA M	1 0 1 0	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA	→TMP
XRI data	1 1 1 0	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2	→TMP
ORA r	1 0 1 1	0 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)+(TPM)→A	
ORA M	1 0 1 1	0 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA	→TMP
ORI data	1 1 1 1	0 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2	→TMP
CMP r	1 0 1 1	1 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)-(TMP), FLAGS	
CMP M	1 0 1 1	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA	→TMP
CPI data	1 1 1 1	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2	→TMP
RLC	0 0 0 0	0 1 1 1				(A)→ALU ROTATE		[9]	ALU→A, CY	
RRC	0 0 0 0	1 1 1 1				(A)→ALU ROTATE		[9]	ALU→A, CY	
RAL	0 0 0 1	0 1 1 1				(A), CY→ALU ROTATE		[9]	ALU→A, CY	
RAR	0 0 0 1	1 1 1 1				(A), CY→ALU ROTATE		[9]	ALU→A, CY	
CMA	0 0 1 0	1 1 1 1				A→ALU COMPLEMENT		[9]	ALU→A	
CMC	0 0 1 1	1 1 1 1				CY→ALU COMPLEMENT		[9]	ALU→CY	
STC	0 0 1 1	0 1 1 1				1→ALU		[9]	ALU→CY	
JMP addr	1 1 0 0	0 0 1 1					X	PC OUT STATUS[6]	PC = PC + 1 B2	→Z
J cond addr[17]	1 1 C C	C 0 1 0				JUDGE CONDITION		PC OUT STATUS[6]	PC = PC + 1 B2	→Z
CALL addr	1 1 0 0	1 1 0 1				SP = SP - 1		PC OUT STATUS[6]	PC = PC + 1 B2	→Z
C cond addr[17]	1 1 C C	C 1 0 0				JUDGE CONDITION IF TRUE, SP = SP - 1		PC OUT STATUS[6]	PC = PC + 1 B2	→Z
RET	1 1 0 0	1 0 0 1					X	SP OUT STATUS[15]	SP = SP + 1 DATA	→PCL
R cond addr[17]	1 1 C C	C 0 0 0			INST-TMP/IR	JUDGE CONDITION[14]		SP OUT STATUS[15]	SP = SP + 1 DATA	→PCL
RST n	1 1 N N	N 1 1 1			φ→W INST-TMP/IR	SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1 (PCH)	→DATA BUS
PCHL	1 1 1 0	1 0 0 1			INST-TMP/IR	(HL) → PC				
PUSH rp	1 1 R P	0 1 0 1				SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1 (rh)	→DATA BUS
PUSH PSW	1 1 1 1	0 1 0 1				SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1 (A)	→DATA BUS
POP rp	1 1 R P	0 0 0 1					X	SP OUT STATUS[15]	SP = SP + 1 DATA	→r/l
POP PSW	1 1 1 1	0 0 0 1					X	SP OUT STATUS[15]	SP = SP + 1 DATA	→FLAGS
XTHL	1 1 1 0	0 0 1 1					X	SP OUT STATUS[15]	SP = SP + 1 DATA	→Z
IN port	1 1 0 1	1 0 1 1					X	PC OUT STATUS[6]	PC = PC + 1 B2	→Z, W
OUT port	1 1 0 1	0 0 1 1					X	PC OUT STATUS[6]	PC = PC + 1 B2	→Z, W
EI	1 1 1 1	1 0 1 1					X	SET INTE F/F [11]		
DI	1 1 1 1	0 0 1 1					X	RESET INTE F/F [11]		
HLT	0 1 1 1	0 1 1 0					X	PC OUT STATUS	HALT MODE[20]	
NOP	0 0 0 0	0 0 0 0	PC OUT STATUS	PC = PC + 1	INST-TMP/IR		X			



M3			M4			M5				
T1	T2[2]	T3	T1	T2[2]	T3	T1	T2[2]	T3	T4	T5
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)-(TMP); FLAGS									
[9]	(ACT)-(TMP); FLAGS									
PC OUT STATUS[6]	PC = PC + 1 B3 → W									WZ OUT STATUS[11] (WZ) + 1 → PC
PC OUT STATUS[6]	PC = PC + 1 B3 → W									WZ OUT STATUS[11,12] (WZ) + 1 → PC
PC OUT STATUS[6]	PC = PC + 1 B3 → W	SP OUT STATUS[16] (PCH) → DATA BUS SP = SP - 1				SP OUT STATUS[16] (PCL) → DATA BUS				WZ OUT STATUS[11] (WZ) + 1 → PC
PC OUT STATUS[6]	PC = PC + 1 B3 → W[13]	SP OUT STATUS[16] (PCH) → DATA BUS SP = SP - 1				SP OUT STATUS[16] (PCL) → DATA BUS				WZ OUT STATUS[11,12] (WZ) + 1 → PC
SP OUT STATUS[15]	SP = SP + 1 DATA → PCH									
SP OUT STATUS[15]	SP = SP + 1 DATA → PCH									
SP OUT STATUS[16]	(TMP = 00NNN000) → Z (PCL) → DATA BUS									WZ OUT STATUS[11] (WZ) + 1 → PC
SP OUT STATUS[16]	(ri) → DATA BUS									
SP OUT STATUS[16]	FLAGS → DATA BUS									
SP OUT STATUS[15]	SP = SP + 1 DATA → rh									
SP OUT STATUS[15]	SP = SP + 1 DATA → A									
SP OUT STATUS[15]	DATA → W	SP OUT STATUS[16] (H) → DATA BUS SP = SP - 1				SP OUT STATUS[16] (L) → DATA BUS			(WZ) → HL	
WZ OUT STATUS[18]	DATA → A									
WZ OUT STATUS[18]	(A) → DATA BUS									

fetch; the first (or only) byte, containing the op code, is fetched during this cycle.

2. If the READY input from memory is not high during T2 of each memory cycle, the processor will enter a wait state (TW) until READY is sampled as high.

3. States T4 and T5 are present, as required, for operations which are completely internal to the CPU. The contents of the internal bus during T4 and T5 are available at the data bus; this is designed for testing purposes only. An "X" denotes that the state is present, but is only used for such internal operations as instruction decoding.

4. Only register pairs  $rp = B$  (registers B and C) or  $rp = D$  (registers D and E) may be specified.

5. These states are skipped.

6. Memory read sub-cycles; an instruction or data word will be read.

7. Memory write sub-cycle.

8. The READY signal is not required during the second and third sub-cycles (M2 and M3). The HOLD signal is accepted during M2 and M3. The SYNC signal is not generated during M2 and M3. During the execution of DAD, M2 and M3 are required for an internal register-pair add; memory is not referenced.

9. The results of these arithmetic, logical or rotate instructions are not moved into the accumulator (A) until state T2 of the next instruction cycle. That is, A is loaded while the next instruction is being fetched; this overlapping of operations allows for faster processing.

10. If the value of the least significant 4-bits of the accumulator is greater than 9 or if the auxiliary carry bit is set, 6 is added to the accumulator. If the value of the most significant 4-bits of the accumulator is now greater than 9, or if the carry bit is set, 6 is added to the most significant 4-bits of the accumulator.

11. This represents the first sub-cycle (the instruction fetch) of the next instruction cycle.

the contents of the program counter (PC).

13. If the condition was not met, sub-cycles M4 and M5 are skipped; the processor instead proceeds immediately to the instruction fetch (M1) of the next instruction cycle.

14. If the condition was not met, sub-cycles M2 and M3 are skipped; the processor instead proceeds immediately to the instruction fetch (M1) of the next instruction cycle.

15. Stack read sub-cycle.

16. Stack write sub-cycle.

17. CONDITION

	CCC
NZ — not zero ( $Z = 0$ )	000
Z — zero ( $Z = 1$ )	001
NC — no carry ( $CY = 0$ )	010
C — carry ( $CY = 1$ )	011
PO — parity odd ( $P = 0$ )	100
PE — parity even ( $P = 1$ )	101
P — plus ( $S = 0$ )	110
M — minus ( $S = 1$ )	111

18. I/O sub-cycle: the I/O port's 8-bit select code is duplicated on address lines 0-7 ( $A_{0-7}$ ) and 8-15 ( $A_{8-15}$ ).

19. Output sub-cycle.

20. The processor will remain idle in the halt state until an interrupt, a reset or a hold is accepted. When a hold request is accepted, the CPU enters the hold mode; after the hold mode is terminated, the processor returns to the halt state. After a reset is accepted, the processor begins execution at memory location zero. After an interrupt is accepted, the processor executes the instruction forced onto the data bus (usually a restart instruction).

SSS or DDD	Value	rp	Value
A	111	B	00
B	000	D	01
C	001	H	10
D	010	SP	11
E	011		
H	100		
L	101		

**Chapter 5**  
**8080A/8085A**  
**Instruction Set**

**MOS**

**6502**

**MOS**

**6502**



## CHAPTER 5 THE INSTRUCTION SET

### 5.1 WHAT THE INSTRUCTION SET IS

A computer, no matter how sophisticated, can do only what it is instructed to do. A program is a sequence of instructions, each of which is recognized by the computer and causes it to perform an operation. Once a program is placed in memory space that is accessible to your CPU, you may run that same sequence of instructions as often as you wish to solve the same problem or to do the same function. The set of instructions to which the 8085A CPU will respond is permanently fixed in the design of the chip.

Each computer instruction allows you to initiate the performance of a specific operation. The 8085A implements a group of instructions that move data between registers, between a register and memory, and between a register and an I/O port. It also has arithmetic and logic instructions, conditional and unconditional branch instructions, and machine control instructions. The CPU recognizes these instructions only when they are coded in binary form.

### 5.2 SYMBOLS AND ABBREVIATIONS:

The following symbols and abbreviations are used in the subsequent description of the 8085A instructions:

SYMBOLS	MEANING
accumulator	Register A
addr	16-bit address quantity
data	8-bit quantity
data 16	16-bit data quantity
byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
port	8-bit address of an I/O device
r,r1,r2	One of the registers A,B,C,D,E,H,L

DDD,SSS

The bit pattern designating one of the registers A,B,C,D,E,H,L (DDD = destination, SSS = source):

DDD or SSS	REGISTER NAME
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp

One of the register pairs:

B represents the B,C pair with B as the high-order register and C as the low-order register;

D represents the D,E pair with D as the high-order register and E as the low-order register;

H represents the H,L pair with H as the high-order register and L as the low-order register;

SP represents the 16-bit stack pointer register.

RP

The bit pattern designating one of the register pairs B,D,H,SP:

RP	REGISTER PAIR
00	B-C
01	D-E
10	H-L
11	SP

rh

The first (high-order) register of a designated register pair.

rl

The second (low-order) register of a designated register pair.



PC	16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively).
SP	16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively).
r <sub>m</sub>	Bit m of the register r (bits are number 7 through 0 from left to right).
LABEL	16-bit address of subroutine.
	The condition flags:
Z	Zero
S	Sign
P	Parity
CY	Carry
AC	Auxiliary Carry
( )	The contents of the memory location or registers enclosed in the parentheses.
←	"Is transferred to"
∧	Logical AND
⊕	Exclusive OR
∨	Inclusive OR
+	Addition
−	Two's complement subtraction
*	Multiplication
↔	"Is exchanged with"
—	The one's complement (e.g., $\overline{A}$ )
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 through 7 respectively.

The instruction set encyclopedia is a detailed description of the 8085A instruction set. Each instruction is described in the following manner:

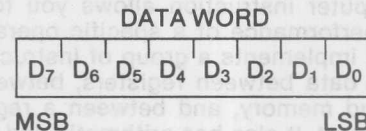
1. The MCS-85 macro assembler format, consisting of the instruction mnemonic and operand fields, is printed in **BOLDFACE** on the first line.
2. The name of the instruction is enclosed in parentheses following the mnemonic.
3. The next lines contain a symbolic description of what the instruction does.
4. This is followed by a narrative description of the operation of the instruction.

5. The boxes describe the binary codes that comprise the machine instruction.
6. The last four lines contain information about the execution of the instruction. The number of machine cycles and states required to execute the instruction are listed first. If the instruction has two possible execution times, as in a conditional jump, both times are listed, separated by a slash. Next, data addressing modes are listed if applicable. The last line lists any of the five flags that are affected by the execution of the instruction.

## 5.3 INSTRUCTION AND DATA FORMATS

Memory used in the MCS-85 system is organized in 8-bit bytes. Each byte has a unique location in physical memory. That location is described by one of a sequence of 16-bit binary addresses. The 8085A can address up to 64K ( $K = 1024$ , or  $2^{10}$ ; hence, 64K represents the decimal number 65,536) bytes of memory, which may consist of both random-access, read-write memory (RAM) and read-only memory (ROM), which is also random-access.

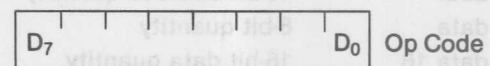
Data in the 8085A is stored in the form of 8-bit binary integers:



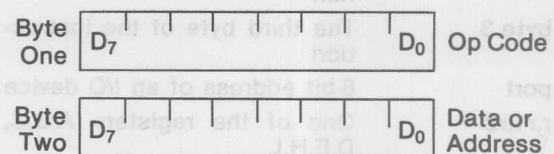
When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the Intel 8085A, BIT 0 is referred to as the **Least Significant Bit (LSB)**, and BIT 7 (of an 8-bit number) is referred to as the **Most Significant Bit (MSB)**.

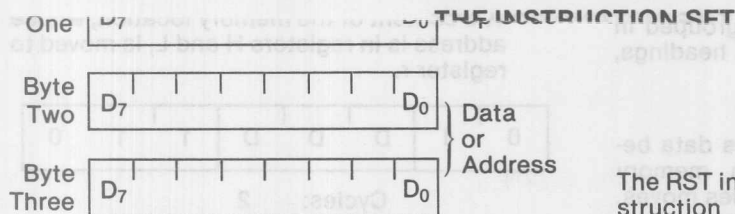
An 8085A program instruction may be one, two or three bytes in length. Multiple-byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instruction. The exact instruction format will depend on the particular operation to be executed.

### Single Byte Instructions



### Two-Byte Instructions





#### 5.4 ADDRESSING MODES:

Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8085A has four different modes for addressing data stored in memory or in registers:

- **Direct** — Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).
- **Register** — The instruction specifies the register or register pair in which the data is located.
- **Register Indirect** — The instruction specifies a register pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair the low-order bits in the second).
- **Immediate** — The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- **Direct** — The branch instruction contains the address of the next instruction to be executed. (Except for the 'RST' instruction, byte 2 contains the low-order address and byte 3 the high-order address.)

the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special one-byte call instruction (usually used during interrupt sequences). RST includes a three-bit field; program control is transferred to the instruction whose address is eight times the contents of this three-bit field.

#### 5.5 CONDITION FLAGS:

There are five condition flags associated with the execution of instructions on the 8085A. They are Zero, Sign, Parity, Carry, and Auxiliary Carry. Each is represented by a 1-bit register (or flip-flop) in the CPU. A flag is set by forcing the bit to 1; it is reset by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner:

**Zero:** If the result of an instruction has the value 0, this flag is set; otherwise it is reset.

**Sign:** If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.

**Parity:** If the modulo 2 sum of the bits of the result of the operation is 0, (i.e., if the result has even parity), this flag is set; otherwise it is reset (i.e., if the result has odd parity).

**Carry:** If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.

**Auxiliary Carry:** If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single-precision additions, subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

## 5.6 INSTRUCTION SET ENCYCLOPEDIA

In the ensuing dozen pages, the complete 8085A instruction set is described, grouped in order under five different functional headings, as follows:

1. **Data Transfer Group** — Moves data between registers or between memory locations and registers. Includes moves, loads, stores, and exchanges. (See below.)
2. **Arithmetic Group** — Adds, subtracts, increments, or decrements data in registers or memory. (See page 5-13.)
3. **Logic Group** — ANDs, ORs, XORs, compares, rotates, or complements data in registers or between memory and a register. (See page 5-16.)
4. **Branch Group** — Initiates conditional or unconditional jumps, calls, returns, and restarts. (See page 5-20.)
5. **Stack, I/O, and Machine Control Group** — Includes instructions for maintaining the stack, reading from input ports, writing to output ports, setting and reading interrupt masks, and setting and clearing flags. (See page 5-22.)

The formats described in the encyclopedia reflect the assembly language processed by Intel-supplied assembler, used with the Intel® development systems.

### 5.6.1 Data Transfer Group

This group of instructions transfers data to and from registers and memory. **Condition flags are not affected by any instruction in this group.**

#### MOV r1, r2 (Move Register)

(r1) ← (r2)

The content of register r2 is moved to register r1.

0	1	D	D	D	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1  
States: 4 (8085), 5 (8080)  
Addressing: register  
Flags: none

#### MOV r, M (Move from memory)

(r) ← ((H) (L))

The content of the memory location, whose address is in registers H and L, is moved to register r.

0	1	D	D	D	1	1	0
---	---	---	---	---	---	---	---

Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: none

#### MOV M, r (Move to memory)

((H) (L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.

0	1	1	1	0	S	S	S
---	---	---	---	---	---	---	---

Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: none

#### MVI r, data (Move Immediate)

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.

0	0	D	D	D	1	1	0
data							

Cycles: 2  
States: 7  
Addressing: immediate  
Flags: none

#### MVI M, data (Move to memory immediate)

((H) (L)) ← (byte 2)

The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.

0	0	1	1	0	1	1	0
data							

Cycles: 3  
States: 10  
Addressing: immed./reg. indirect  
Flags: none

## LXI rp, data 16 (Load register pair immediate)

(rh) ← (byte 3),

(rl) ← (byte 2)

Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.

0	0	R	P	0	0	0	1
low-order data							
high-order data							

Cycles: 3  
States: 10  
Addressing: immediate  
Flags: none

## LDA addr (Load Accumulator direct)

(A) ← ((byte 3)(byte 2))

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.

0	0	1	1	1	0	1	0
low-order addr							
high-order addr							

Cycles: 4  
States: 13  
Addressing: direct  
Flags: none

## STA addr (Store Accumulator direct)

((byte 3)(byte 2)) ← (A)

The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.

0	0	1	1	0	0	1	0
low-order addr							
high-order addr							

Cycles: 4  
States: 13  
Addressing: direct  
Flags: none

## LHLD addr (Load H and L direct)

(L) ← ((byte 3)(byte 2))

(H) ← ((byte 3)(byte 2) + 1)

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.

0	0	1	0	1	0	1	0
low-order addr							
high-order addr							

Cycles: 5  
States: 16  
Addressing: direct  
Flags: none

## SHLD addr (Store H and L direct)

((byte 3)(byte 2)) ← (L)

((byte 3)(byte 2) + 1) ← (H)

The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.

0	0	1	0	0	0	1	0
low-order addr							
high-order addr							

Cycles: 5  
States: 16  
Addressing: direct  
Flags: none

## LDAX rp (Load accumulator indirect)

(A) ← ((rp))

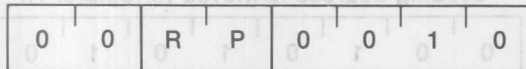
The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

0	0	R	P	1	0	1	0
---	---	---	---	---	---	---	---

Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: none

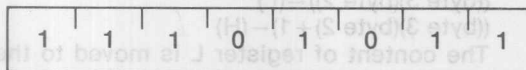
## THE INSTRUCTION SET

**STAX rp** (Store accumulator indirect)  
 $((rp)) \leftarrow (A)$   
 The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs  $rp = B$  (registers B and C) or  $rp = D$  (registers D and E) may be specified.



Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: none

**XCHG** (Exchange H and L with D and E)  
 $(H) \leftrightarrow (D)$   
 $(L) \leftrightarrow (E)$   
 The contents of registers H and L are exchanged with the contents of registers D and E.



Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: none

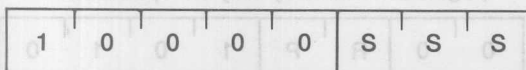
### 5.6.2 Arithmetic Group

This group of instructions performs arithmetic operations on data in registers and memory.

**Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.**

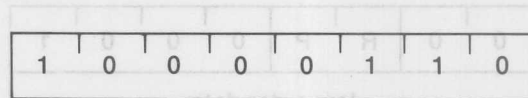
All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

**ADD r** (Add Register)  
 $(A) \leftarrow (A) + (r)$   
 The content of register r is added to the content of the accumulator. The result is placed in the accumulator.



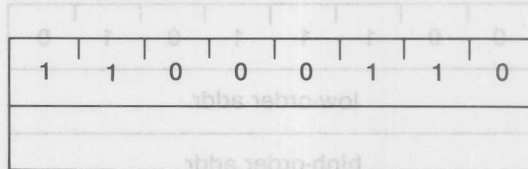
Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

**ADD M** (Add memory)  
 $(A) \leftarrow (A) + ((H) (L))$   
 The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.



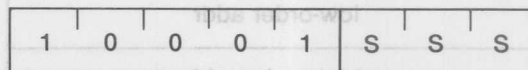
Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**ADI data** (Add immediate)  
 $(A) \leftarrow (A) + (\text{byte } 2)$   
 The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**ADC r** (Add Register with carry)  
 $(A) \leftarrow (A) + (r) + (CY)$   
 The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.

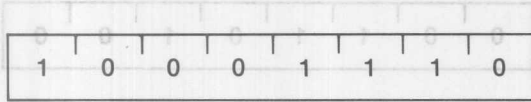


Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC



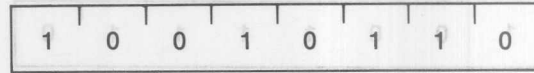
## THE INSTRUCTION SET

**ADC M** (Add memory with carry)  
 $(A) \leftarrow (A) + ((H) (L)) + (CY)$   
 The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.



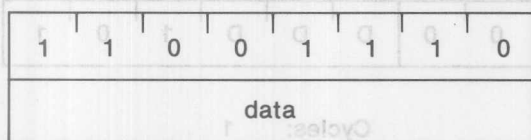
Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**SUB M** (Subtract memory)  
 $(A) \leftarrow (A) - ((H) (L))$   
 The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.



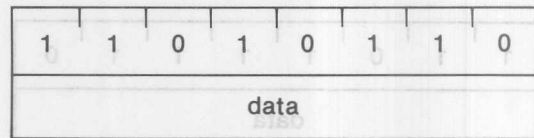
Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**ACI data** (Add immediate with carry)  
 $(A) \leftarrow (A) + (\text{byte 2}) + (CY)$   
 The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.



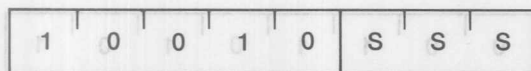
Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**SUI data** (Subtract immediate)  
 $(A) \leftarrow (A) - (\text{byte 2})$   
 The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.



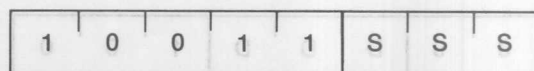
Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**SUB r** (Subtract Register)  
 $(A) \leftarrow (A) - (r)$   
 The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.



Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

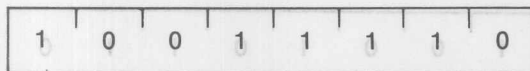
**SBB r** (Subtract Register with borrow)  
 $(A) \leftarrow (A) - (r) - (CY)$   
 The content of register r and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

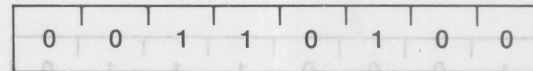
## THE INSTRUCTION SET

**SBB M** (Subtract memory with borrow)  
 $(A) \leftarrow (A) - ((H) (L)) - (CY)$   
 The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



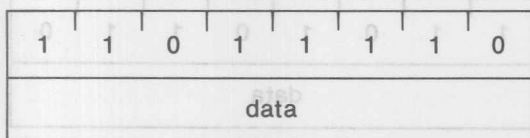
Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**INR M** (Increment memory)  
 $((H) (L)) \leftarrow ((H) (L)) + 1$   
 The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags **except** CY are affected.



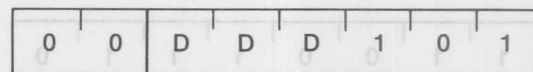
Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: Z,S,P,AC

**SBI data** (Subtract immediate with borrow)  
 $(A) \leftarrow (A) - (\text{byte 2}) - (CY)$   
 The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



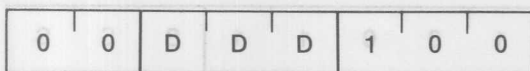
Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**DCR r** (Decrement Register)  
 $(r) \leftarrow (r) - 1$   
 The content of register r is decremented by one. Note: All condition flags **except** CY are affected.



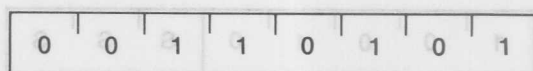
Cycles: 1  
 States: 4 (8085), 5 (8080)  
 Addressing: register  
 Flags: Z,S,P,AC

**INR r** (Increment Register)  
 $(r) \leftarrow (r) + 1$   
 The content of register r is incremented by one. Note: All condition flags **except** CY are affected.



Cycles: 1  
 States: 4 (8085), 5 (8080)  
 Addressing: register  
 Flags: Z,S,P,AC

**DCR M** (Decrement memory)  
 $((H) (L)) \leftarrow ((H) (L)) - 1$   
 The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags **except** CY are affected.



Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: Z,S,P,AC

## THE INSTRUCTION SET

cremented by one. Note: **No condition flags are affected.**

0	0	R	P	0	0	1	1
---	---	---	---	---	---	---	---

Cycles: 1  
States: 6 (8085), 5 (8080)  
Addressing: register  
Flags: none

### DCX rp (Decrement register pair)

(rh) (rl)  $\leftarrow$  (rh) (rl) - 1

The content of the register pair rp is decremented by one. Note: **No condition flags are affected.**

0	0	R	P	1	0	1	1
---	---	---	---	---	---	---	---

Cycles: 1  
States: 6 (8085), 5 (8080)  
Addressing: register  
Flags: none

### DAD rp (Add register pair to H and L)

(H) (L)  $\leftarrow$  (H) (L) + (rh) (rl)

The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: **Only the CY flag is affected.** It is set if there is a carry out of the double precision add; otherwise it is reset.

0	0	R	P	1	0	0	1
---	---	---	---	---	---	---	---

Cycles: 3  
States: 10  
Addressing: register  
Flags: CY

Decimal digits by the following process:

1. If the value of the least significant 4 bits of the accumulator is greater than 9 or if the AC flag is set, 6 is added to the accumulator.
2. If the value of the most significant 4 bits of the accumulator is now greater than 9, or if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.

0	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---

Cycles: 1  
States: 4  
Flags: Z,S,P,CY,AC

### 5.6.3 Logical Group

This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

### ANA r (AND Register)

(A)  $\leftarrow$  (A)  $\wedge$  (r)

The content of register r is logically ANDed with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set (8085). The CY flag is cleared and AC is set to the OR'ing of bits 3 of the operands (8080).**

1	0	1	0	0	S	S	S
---	---	---	---	---	---	---	---

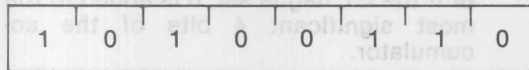
Cycles: 1  
States: 4  
Addressing: register  
Flags: Z,S,P,CY,AC

## THE INSTRUCTION SET

### ANA M (AND memory)

$$(A) \leftarrow (A) \wedge ((H) (L))$$

The contents of the memory location whose address is contained in the H and L registers is logically ANDed with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set (8085). The CY flag is cleared and AC is set to the OR'ing of bits 3 of the operands (8080).**

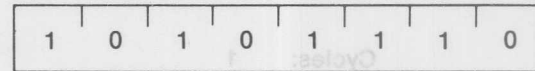


Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

### XRA M (Exclusive OR Memory)

$$(A) \leftarrow (A) \vee ((H) (L))$$

The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

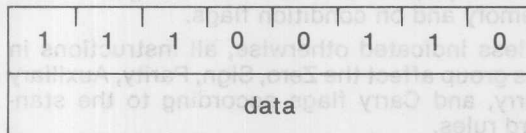


Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

### ANI data (AND immediate)

$$(A) \leftarrow (A) \wedge (\text{byte } 2)$$

The content of the second byte of the instruction is logically ANDed with the contents of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set (8085). The CY flag is cleared and AC is set to the OR'ing of bits 3 of the operands (8080).**

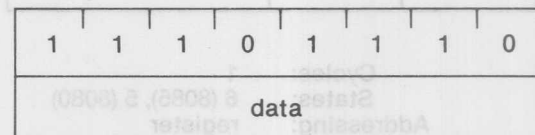


Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

### XRI data (Exclusive OR immediate)

$$(A) \leftarrow (A) \vee (\text{byte } 2)$$

The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

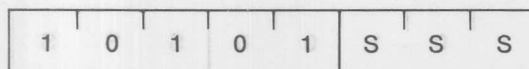


Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

### XRA r (Exclusive OR Register)

$$(A) \leftarrow (A) \vee (r)$$

The content of register r is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

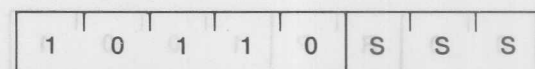


Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

### ORA r (OR Register)

$$(A) \leftarrow (A) \vee (r)$$

The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

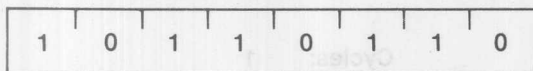


Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

## ORA M (OR memory)

(A) ← (A) V ((H) (L))

The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

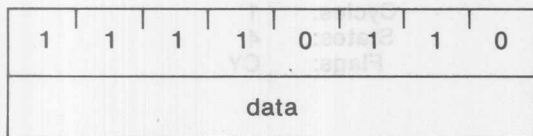


Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: Z,S,P,CY,AC

## ORI data (OR Immediate)

(A) ← (A) V (byte 2)

The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

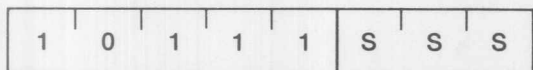


Cycles: 2  
States: 7  
Addressing: immediate  
Flags: Z,S,P,CY,AC

## CMP r (Compare Register)

(A) ← (r)

The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = (r). The CY flag is set to 1 if (A) < (r).**

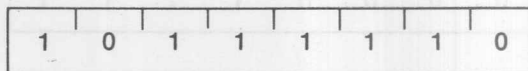


Cycles: 1  
States: 4  
Addressing: register  
Flags: Z,S,P,CY,AC

## CMP M (Compare memory)

(A) ← ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = ((H) (L)). The CY flag is set to 1 if (A) < ((H) (L)).**

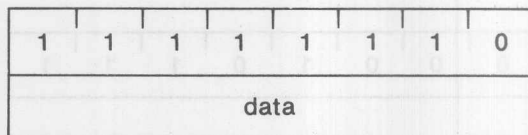


Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: Z,S,P,CY,AC

## CPI data (Compare immediate)

(A) ← (byte 2)

The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. **The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A) < (byte 2).**



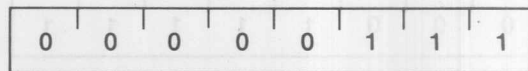
Cycles: 2  
States: 7  
Addressing: immediate  
Flags: Z,S,P,CY,AC

## RLC (Rotate left)

(A<sub>n+1</sub>) ← (A<sub>n</sub>); (A<sub>0</sub>) ← (A<sub>7</sub>)

(CY) ← (A<sub>7</sub>)

The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position. **Only the CY flag is affected.**



Cycles: 1  
States: 4  
Flags: CY

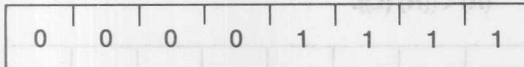


## THE INSTRUCTION SET

### RRC (Rotate right)

$(A_n) \leftarrow (A_{n+1}); (A_7) \leftarrow (A_0)$   
 $(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. **Only the CY flag is affected.**

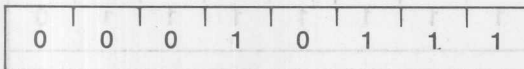


Cycles: 1  
 States: 4  
 Flags: CY

### RAL (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n); (CY) \leftarrow (A_7)$   
 $(A_0) \leftarrow (CY)$

The content of the accumulator is rotated left one position through the CY flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. **Only the CY flag is affected.**

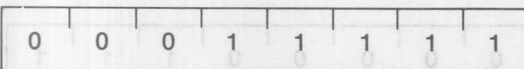


Cycles: 1  
 States: 4  
 Flags: CY

### RAR (Rotate right through carry)

$(A_n) \leftarrow (A_{n+1}); (CY) \leftarrow (A_0)$   
 $(A_7) \leftarrow (CY)$

The content of the accumulator is rotated right one position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit. **Only the CY flag is affected.**

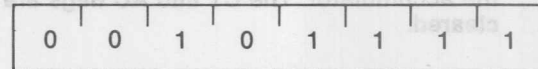


Cycles: 1  
 States: 4  
 Flags: CY

### CMA (Complement accumulator)

$(A) \leftarrow (\bar{A})$

The contents of the accumulator are complemented (zero bits become 1, one bits become 0). **No flags are affected.**

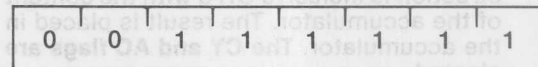


Cycles: 1  
 States: 4  
 Flags: none

### CMC (Complement carry)

$(CY) \leftarrow (\bar{CY})$

The CY flag is complemented. **No other flags are affected.**

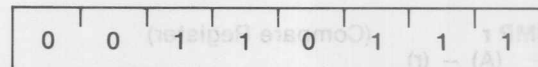


Cycles: 1  
 States: 4  
 Flags: CY

### STC (Set carry)

$(CY) \leftarrow 1$

The CY flag is set to 1. **No other flags are affected.**



Cycles: 1  
 States: 4  
 Flags: CY

## 5.6.4 Branch Group

This group of instructions alter normal sequential program flow.

**Condition flags are not affected** by any instruction in this group.

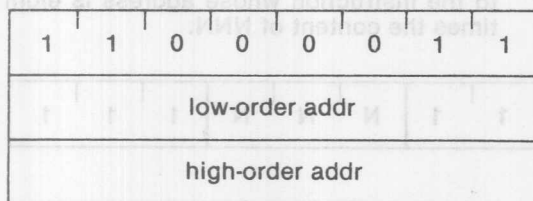
The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

CONDITION	CCC
NZ — not zero (Z = 0)	000
Z — zero (Z = 1)	001
NC — no carry (CY = 0)	010
C — carry (CY = 1)	011
PO — parity odd (P = 0)	100
PE — parity even (P = 1)	101
P — plus (S = 0)	110
M — minus (S = 1)	111

### JMP addr (Jump)

(PC) ← (byte 3) (byte 2)

Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



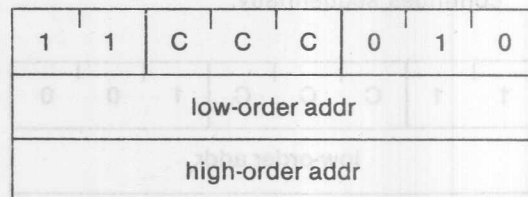
Cycles: 3  
States: 10  
Addressing: immediate  
Flags: none

### Jcondition addr (Conditional jump)

If (CCC),

(PC) ← (byte 3) (byte 2)

If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.



Cycles: 2/3 (8085), 3 (8080)

States: 7/10 (8085), 10 (8080)

Addressing: immediate

Flags: none

### CALL addr (Call)

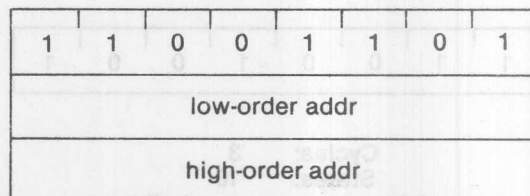
((SP) - 1) ← (PCH)

((SP) - 2) ← (PCL)

(SP) ← (SP) - 2

(PC) ← (byte 3) (byte 2)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



Cycles: 5

States: 18 (8085), 17 (8080)

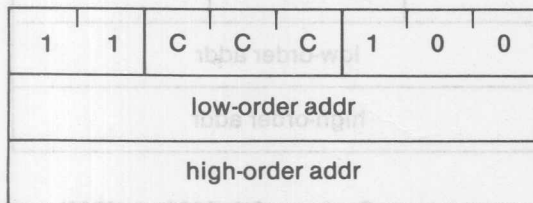
Addressing: immediate/  
reg. indirect

Flags: none

## THE INSTRUCTION SET

### Ccondition addr (Condition call)

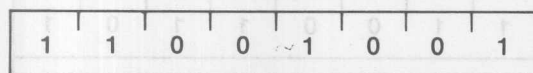
If (CCC),  
 $((SP) - 1) \leftarrow (PCH)$   
 $((SP) - 2) \leftarrow (PCL)$   
 $(SP) \leftarrow (SP) - 2$   
 $(PC) \leftarrow (\text{byte 3}) (\text{byte 2})$   
 If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.



Cycles: 2/5 (8085), 3/5 (8080)  
 States: 9/18 (8085), 11/17 (8080)  
 Addressing: immediate/  
 reg. indirect  
 Flags: none

### RET (Return)

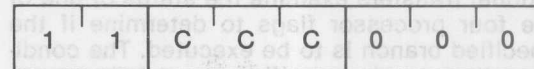
$(PCL) \leftarrow ((SP));$   
 $(PCH) \leftarrow ((SP) + 1);$   
 $(SP) \leftarrow (SP) + 2;$   
 The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.



Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: none

### Rcondition (Conditional return)

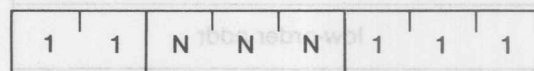
If (CCC),  
 $(PCL) \leftarrow ((SP))$   
 $(PCH) \leftarrow ((SP) + 1)$   
 $(SP) \leftarrow (SP) + 2$   
 If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.



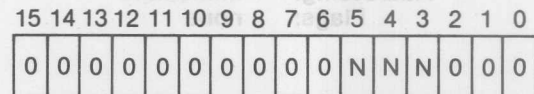
Cycles: 1/3  
 States: 6/12 (8085), 5/11 (8080)  
 Addressing: reg. indirect  
 Flags: none

### RST n (Restart)

$((SP) - 1) \leftarrow (PCH)$   
 $((SP) - 2) \leftarrow (PCL)$   
 $(SP) \leftarrow (SP) - 2$   
 $(PC) \leftarrow 8 * (NNN)$   
 The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

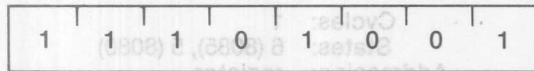


Cycles: 3  
 States: 12 (8085), 11 (8080)  
 Addressing: reg. indirect  
 Flags: none



Program Counter After Restart

**PCHL** (Jump H and L indirect — move H and L to PC) —  
 (PCH) ← (H)  
 (PCL) ← (L)  
 The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.



Cycles: 1  
 States: 6 (8085), 5 (8080)  
 Addressing: register  
 Flags: none

## 5.6.5 Stack, I/O, and Machine Control Group

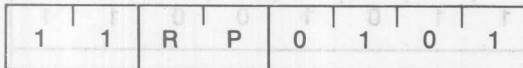
This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, **condition flags are not affected by any instructions in this group.**

**PUSH rp** (Push)

((SP) - 1) ← (rh)  
 ((SP) - 2) ← (rl)  
 ((SP) - (SP) - 2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. **Note: Register pair rp = SP may not be specified.**

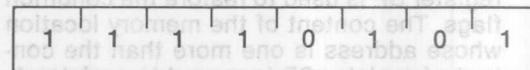


Cycles: 3  
 States: 12 (8085), 11 (8080)  
 Addressing: reg. indirect  
 Flags: none

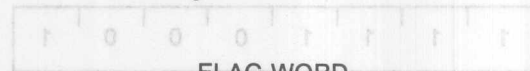
**PUSH PSW** (Push processor status word)

((SP) - 1) ← (A)  
 ((SP) - 2)<sub>0</sub> ← (CY), ((SP) - 2)<sub>1</sub> ← X  
 ((SP) - 2)<sub>2</sub> ← (P), ((SP) - 2)<sub>3</sub> ← X  
 ((SP) - 2)<sub>4</sub> ← (AC), ((SP) - 2)<sub>5</sub> ← X  
 ((SP) - 2)<sub>6</sub> ← (Z), ((SP) - 2)<sub>7</sub> ← (S)  
 ((SP) - (SP) - 2 X: Undefined.

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.



Cycles: 3  
 States: 12 (8085), 11 (8080)  
 Addressing: reg. indirect  
 Flags: none



FLAG WORD

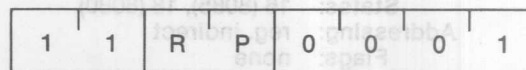
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
S	Z	X	AC	X	P	X	CY

X: undefined

**POP rp** (Pop)

(rl) ← ((SP))  
 (rh) ← ((SP) + 1)  
 (SP) ← (SP) + 2

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register rp. The content of register SP is incremented by 2. **Note: Register pair rp = SP may not be specified.**



Cycles: 3  
 States: 10  
 Addressing: reg.indirect  
 Flags: none

## THE INSTRUCTION SET

### POP PSW (Pop processor status word)

(CY) ← ((SP))<sub>0</sub>

(P) ← ((SP))<sub>2</sub>

(AC) ← ((SP))<sub>4</sub>

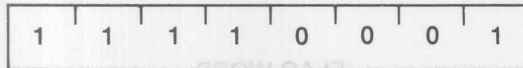
(Z) ← ((SP))<sub>6</sub>

(S) ← ((SP))<sub>7</sub>

(A) ← ((SP) + 1)

(SP) ← (SP) + 2

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.



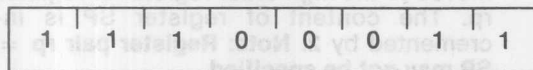
Cycles: 3  
States: 10  
Addressing: reg. indirect  
Flags: Z,S,P,CY,AC

### XTHL (Exchange stack top with H and L)

(L) ↔ ((SP))

(H) ↔ ((SP) + 1)

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

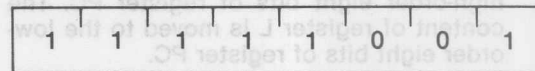


Cycles: 5  
States: 16 (8085), 18 (8080)  
Addressing: reg. indirect  
Flags: none

### SPHL (Move HL to SP)

(SP) ← (H) (L)

The contents of registers H and L (16 bits) are moved to register SP.

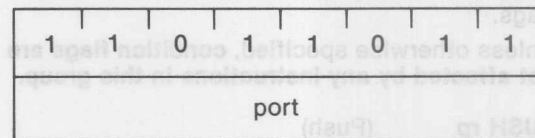


Cycles: 1  
States: 6 (8085), 5 (8080)  
Addressing: register  
Flags: none

### IN port (Input)

(A) ← (data)

The data placed on the eight bit bi-directional data bus by the specified port is moved to register A.

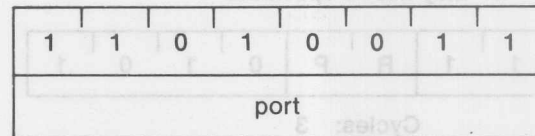


Cycles: 3  
States: 10  
Addressing: direct  
Flags: none

### OUT port (Output)

(data) ← (A)

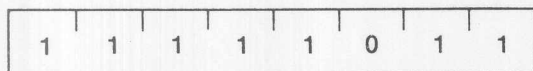
The content of register A is placed on the eight bit bi-directional data bus for transmission to the specified port.



Cycles: 3  
States: 10  
Addressing: direct  
Flags: none



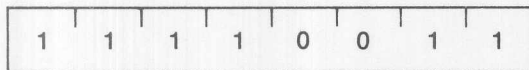
**EI** (Enable interrupts)  
The interrupt system is enabled **following the execution of the next instruction. Interrupts are not recognized during the EI instruction.**



Cycles: 1  
States: 4  
Flags: none

NOTE: Placing an EI instruction on the bus in response to  $\overline{INTA}$  during an  $\overline{INA}$  cycle is prohibited. (8085)

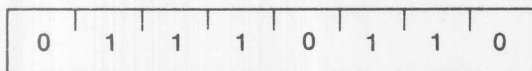
**DI** (Disable interrupts)  
The interrupt system is disabled **immediately following the execution of the DI instruction. Interrupts are not recognized during the DI instruction.**



Cycles: 1  
States: 4  
Flags: none

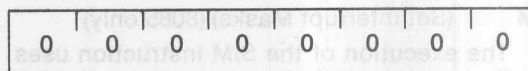
NOTE: Placing a DI instruction on the bus in response to  $\overline{INTA}$  during an  $\overline{INA}$  cycle is prohibited. (8085)

**HLT** (Halt)  
The processor is stopped. The registers and flags are unaffected. (8080) A second ALE is generated during the execution of HLT to strobe out the Halt cycle status information. (8085)



Cycles: 1 + (8085), 1 (8080)  
States: 5 (8085), 7 (8080)  
Flags: none

**NOP** (No op)  
No operation is performed. The registers and flags are unaffected.



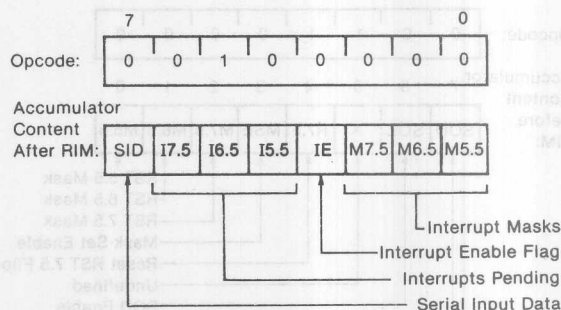
Cycles: 1  
States: 4  
Flags: none

**RIM** (Read Interrupt Masks) (8085 only)

The RIM instruction loads data into the accumulator relating to interrupts and the serial input. This data contains the following information:

- Current interrupt mask status for the RST 5.5, 6.5, and 7.5 hardware interrupts (1 = mask disabled)
- Current interrupt enable flag status (1 = interrupts enabled) except immediately following a TRAP interrupt. (See below.)
- Hardware interrupts pending (i.e., signal received but not yet serviced), on the RST 5.5, 6.5, and 7.5 lines.
- Serial input data.

Immediately following a TRAP interrupt, the RIM instruction must be executed as a part of the service routine if you need to retrieve current interrupt status later. Bit 3 of the accumulator is (in this special case only) loaded with the interrupt enable (IE) flag status that existed prior to the TRAP interrupt. Following an RST 5.5, 6.5, 7.5, or INTR interrupt, the interrupt flag flip-flop reflects the current interrupt enable status. Bit 6 of the accumulator (I7.5) is loaded with the status of the RST 7.5 flip-flop, which is always set (edge-triggered) by an input on the RST 7.5 input line, even when that interrupt has been previously masked. (See SIM Instruction.)



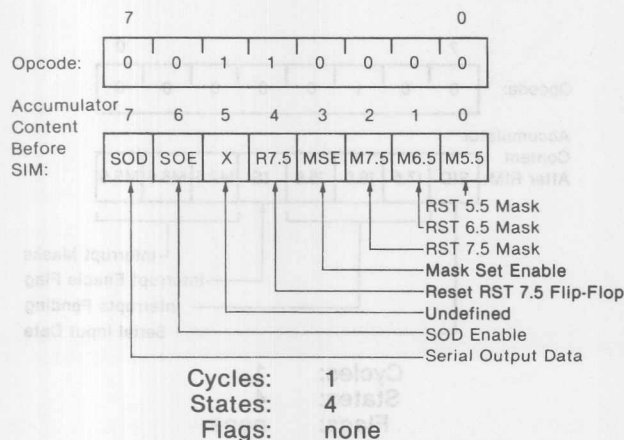
Cycles: 1  
States: 4  
Flags: none

the contents of the accumulator (which must be previously loaded) to perform the following functions:

- Program the interrupt mask for the RST 5.5, 6.5, and 7.5 hardware interrupts.
- Reset the edge-triggered RST 7.5 input latch.
- Load the SOD output latch.

To program the interrupt masks, first set accumulator bit 3 to 1 and set to 1 any bits 0, 1, and 2, which disable interrupts RST 5.5, 6.5, and 7.5, respectively. Then do a SIM instruction. If accumulator bit 3 is 0 when the SIM instruction is executed, the interrupt mask register will not change. If accumulator bit 4 is 1 when the SIM instruction is executed, the RST 7.5 latch is then reset. RST 7.5 is distinguished by the fact that its latch is always set by a rising edge on the RST 7.5 input pin, even if the jump to service routine is inhibited by masking. This latch remains high until cleared by a RESET IN, by a SIM Instruction with accumulator bit 4 high, or by an internal processor acknowledge to an RST 7.5 interrupt subsequent to the removal of the mask (by a SIM instruction). The RESET IN signal always sets all three RST mask bits.

If accumulator bit 6 is at the 1 level when the SIM instruction is executed, the state of accumulator bit 7 is loaded into the SOD latch and thus becomes available for interface to an external device. The SOD latch is unaffected by the SIM instruction if bit 6 is 0. SOD is always reset by the RESET IN signal.



## 8080A/8085A INSTRUCTION SET INDEX

Table 5-1

Instruction	Code	Bytes	T States		Machine Cycles
			8085A	8080A	
ACI DATA	CE data	2	7	7	FR
ADC REG	1000 1SSS	1	4	4	F
ADC M	8E	1	7	7	FR
ADD REG	1000 0SSS	1	4	4	F
ADD M	86	1	7	7	FR
ADI DATA	C6 data	2	7	7	FR
ANA REG	1010 0SSS	1	4	4	F
ANA M	A6	1	7	7	FR
ANI DATA	E6 data	2	7	7	FR
CALL LABEL	CD addr	3	18	17	SR RWW*
CC LABEL	DC addr	3	9/18	11/17	SR*/SR RWW*
CM LABEL	FC addr	3	9/18	11/17	SR*/SR RWW*
CMA	2F	1	4	4	F
CMC	3F	1	4	4	F
CMP REG	1011 1SSS	1	4	4	F
CMP M	BE	1	7	7	FR
CNC LABEL	D4 addr	3	9/18	11/17	SR*/SR RWW*
CNZ LABEL	C4 addr	3	9/18	11/17	SR*/SR RWW*
CP LABEL	F4 addr	3	9/18	11/17	SR*/SR RWW*
CPE LABEL	EC addr	3	9/18	11/17	SR*/SR RWW*
CPI DATA	FE data	2	7	7	FR
CPO LABEL	E4 addr	3	9/18	11/17	SR*/SR RWW*
CZ LABEL	CC addr	3	9/18	11/17	SR*/SR RWW*
DAA	27	1	4	4	F
DAD RP	00RP 1001	1	10	10	FBB
DCR REG	00SS S101	1	4	5	F*
DCR M	35	1	10	10	FRW
DCX RP	00RP 1011	1	6	5	S*
DI	F3	1	4	4	F
EI	FB	1	4	4	F
HLT	76	1	5	7	FB
IN PORT	DB data	2	10	10	FR I
INR REG	00SS S100	1	4	5	F*
INR M	34	1	10	10	FRW
INX RP	00RP 0011	1	6	5	S*
JC LABEL	DA addr	3	7/10	10	FR/F R R†
JM LABEL	FA addr	3	7/10	10	FR/F R R†
JMP LABEL	C3 addr	3	10	10	FR R
JNC LABEL	D2 addr	3	7/10	10	FR/F R R†
JNZ LABEL	C2 addr	3	7/10	10	FR/F R R†
JP LABEL	F2 addr	3	7/10	10	FR/F R R†
JPE LABEL	EA addr	3	7/10	10	FR/F R R†
JPO LABEL	E2 addr	3	7/10	10	FR/F R R†
JZ LABEL	CA addr	3	7/10	10	FR/F R R†
LDA ADDR	3A addr	3	13	13	FR R R
LDAX RP	000X 1010	1	7	7	FR
LHLD ADDR	2A addr	3	16	16	FR R R R
LXI RP, DATA16	00RP 0001 data16	3	10	10	FR R R
MOV REG, REG	01DD DSSS	1	4	5	F*
MOV M, REG	0111 0SSS	1	7	7	FRW
MOV REG, M	01DD D110	1	7	7	FR
MVI REG, DATA	00DD D110 data	2	7	7	FR
MVI M, DATA	36 data	2	10	10	FRW
NOP	00	1	4	4	F
ORA REG	1011 0SSS	1	4	4	F
ORA M	B6	1	7	7	FR
ORI DATA	F6 data	2	7	7	FR
OUT PORT	D3 data	2	10	10	FR O
PCHL	E9	1	6	5	S*
POP RP	11RP 0001	1	10	10	FR R R
PUSH RP	11RP 0101	1	12	11	SRW*
RAL	17	1	4	4	F
RAR	1F	1	4	4	F
RC	D8	1	6/12	5/11	S/S R R*
RET	C9	1	10	10	FR R
RIM (8085A only)	20	1	4	—	F
RLC	07	1	4	4	F
RM	F8	1	6/12	5/11	S/S R R*
RNC	D0	1	6/12	5/11	S/S R R*
RNZ	C0	1	6/12	5/11	S/S R R*
RP	F0	1	6/12	5/11	S/S R R*
RPE	E8	1	6/12	5/11	S/S R R*
RPO	E0	1	6/12	5/11	S/S R R*
RRC	0F	1	4	4	F
RST N	11XX X111	1	12	11	SRW*
RZ	C8	1	6/12	5/11	S/S R R*
SBB REG	1001 1SSS	1	4	4	F
SBB M	9E	1	7	7	FR
SBI DATA	DE data	2	7	7	FR
SHLD ADDR	22 addr	3	16	16	FR R W W
SIM (8085A only)	30	1	4	—	F
SPHL	F9	1	6	5	S*
STA ADDR	32 addr	3	13	13	FR R W
STAX RP	000X 0010	1	7	7	FRW
STC	37	1	4	4	F
SUB REG	1001 0SSS	1	4	4	F
SUB M	96	1	7	7	FR
SUI DATA	D6 data	2	7	7	FR
XCHG	EB	1	4	4	F
XRA REG	1010 1SSS	1	4	4	F
XRA M	AE	1	7	7	FR
XRI DATA	EE data	2	7	7	FR
XTHL	E3	1	16	18	FR R W W

Machine cycle types:

F Four clock period instr fetch

S Six clock period instr fetch

R Memory read

I I/O read

W Memory write

O I/O write

B Bus idle

X Variable or optional binary digit

DDD Binary digits identifying a destination register B = 000, C = 001, D = 010 Memory = 110

SSS Binary digits identifying a source register E = 011, H = 100, L = 101 A = 111

RP Register Pair BC = 00, HL = 10  
DE = 01, SP = 11

\*Five clock period instruction fetch with 8080A.

†The longer machine cycle sequence applies regardless of condition evaluation with 8080A.

•An extra READ cycle (R) will occur for this condition with 8080A.

## 8085A CPU INSTRUCTIONS IN OPERATION CODE SEQUENCE

Table 5-2

OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC
00	NOP	2B	DCX H	56	MOV D,M	81	ADD C	AC	XRA H	D7	RST 2
01	LXI B,D16	2C	INR L	57	MOV D,A	82	ADD D	AD	XRA L	D8	RC
02	STAX B	2D	DCR L	58	MOV E,B	83	ADD E	AE	XRA M	D9	—
03	INX B	2E	MVI L,D8	59	MOV E,C	84	ADD H	AF	XRA A	DA	JC Adr
04	INR B	2F	CMA	5A	MOV E,D	85	ADD L	B0	ORA B	DB	IN D8
05	DCR B	30	SIM	5B	MOV E,H	86	ADD M	B1	ORA C	DC	CC Adr
06	MVI B,D8	31	LXI SP,D16	5C	MOV E,H	87	ADD A	B2	ORA D	DD	—
07	RLC	32	STA Adr	5D	MOV E,L	88	ADC B	B3	ORA E	DE	SBI D8
08	—	33	INX SP	5E	MOV E,M	89	ADC C	B4	ORA H	DF	RST 3
09	DAD B	34	INR M	5F	MOV E,A	8A	ADC D	B5	ORA L	E0	RPO
0A	LDAX B	35	DCR M	60	MOV H,B	8B	ADC E	B6	ORA M	E1	POP H
0B	DCX B	36	MVI M,D8	61	MOV H,C	8C	ADC H	B7	ORA A	E2	JPO Adr
0C	INR C	37	STC	62	MOV H,D	8D	ADC L	B8	CMP B	E3	XTHL
0D	DCR C	38	—	63	MOV H,E	8E	ADC M	B9	CMP C	E4	CPO Adr
0E	MVI C,D8	39	DAD SP	64	MOV H,H	8F	ADC A	BA	CMP D	E5	PUSH H
0F	RRC	3A	LDA Adr	65	MOV H,L	90	SUB B	BB	CMP E	E6	ANI D8
10	—	3B	DCX SP	66	MOV H,M	91	SUB C	BC	CMP H	E7	RST 4
11	LXI D,D16	3C	INR A	67	MOV H,A	92	SUB D	BD	CMP L	E8	RPE
12	STAX D	3D	DCR A	68	MOV L,B	93	SUB E	BE	CMP M	E9	PCHL
13	INX D	3E	MVI A,D8	69	MOV L,C	94	SUB H	BF	CMP A	EA	JPE Adr
14	INR D	3F	CMC	6A	MOV L,D	95	SUB L	C0	RNZ	EB	XCHG
15	DCR D	40	MOV B,B	6B	MOV L,E	96	SUB M	C1	POP B	EC	CPE Adr
16	MVI D,D8	41	MOV B,C	6C	MOV L,H	97	SUB A	C2	JNZ Adr	ED	—
17	RAL	42	MOV B,D	6D	MOV L,L	98	SBB B	C3	JMP Adr	EE	XRI D8
18	—	43	MOV B,E	6E	MOV L,M	99	SBB C	C4	CNZ Adr	EF	RST 5
19	DAD D	44	MOV B,H	6F	MOV L,A	9A	SBB D	C5	PUSH B	F0	RP
1A	LDAX D	45	MOV B,L	70	MOV M,B	9B	SBB E	C6	ADI D8	F1	POP PSW
1B	DCX D	46	MOV B,M	71	MOV M,C	9C	SBB H	C7	RST 0	F2	JP Adr
1C	INR E	47	MOV B,A	72	MOV M,D	9D	SBB L	C8	RZ	F3	DI
1D	DCR E	48	MOV C,B	73	MOV M,E	9E	SBB M	C9	RET Adr	F4	CP Adr
1E	MVI E,D8	49	MOV C,C	74	MOV M,H	9F	SBB A	CA	JZ	F5	PUSH PSW
1F	RAR	4A	MOV C,D	75	MOV M,L	A0	ANA B	CB	—	F6	ORI D8
20	RIM	4B	MOV C,E	76	HLT	A1	ANA C	CC	CZ Adr	F7	RST 6
21	LXI H,D16	4C	MOV C,H	77	MOV M,A	A2	ANA D	CD	CALL Adr	F8	RM
22	SHLD Adr	4D	MOV C,L	78	MOV A,B	A3	ANA E	CE	ACI D8	F9	SPHL
23	INX H	4E	MOV C,M	79	MOV A,C	A4	ANA H	CF	RST 1	FA	JM Adr
24	INR H	4F	MOV C,A	7A	MOV A,D	A5	ANA L	D0	RNC	FB	EI
25	DCR H	50	MOV D,B	7B	MOV A,E	A6	ANA M	D1	POP D	FC	CM Adr
26	MVI H,D8	51	MOV D,C	7C	MOV A,H	A7	ANA A	D2	JNC Adr	FD	—
27	DAA	52	MOV D,D	7D	MOV A,L	A8	XRA B	D3	OUT D8	FE	CPI D8
28	—	53	MOV D,E	7E	MOV A,M	A9	XRA C	D4	CNC Adr	FF	RST 7
29	DAD H	54	MOV D,H	7F	MOV A,A	AA	XRA D	D5	PUSH D		
2A	LHLD Adr	55	MOV D,L	80	ADD B	AB	XRA E	D6	SUI D8		

D8 = constant, or logical/arithmetic expression that evaluates to an 8-bit data quantity.

Adr = 16-bit address.

D16 = constant, or logical/arithmetic expression that evaluates to a 16-bit data quantity.

**8085A INSTRUCTION SET SUMMARY BY FUNCTIONAL GROUPING**  
**Table 5-3**

Instruction Code (1)											Instruction Code (1)											
Mnemonic	Description	D7	D6	D5	D4	D3	D2	D1	D0	Page	Mnemonic	Description	D7	D6	D5	D4	D3	D2	D1	D0	Page	
MOVE, LOAD, AND STORE																						
MOV r1 r2	Move register to register	0	1		D	D	D	S	S	S	5-4	CZ	Call on zero	1	1	0	0	1	1	0	0	5-14
MOV M.r	Move register to memory	0	1	1	1	0	S	S	S	5-4	CNZ	Call on no zero	1	1	0	0	0	1	0	0	5-14	
MOV r.M	Move memory to register	0	1		D	D	D	1	1	0	5-4	CP	Call on positive	1	1	1	1	0	1	0	0	5-14
MVI r	Move immediate register	0	0		D	D	D	1	1	0	5-4	CM	Call on minus	1	1	1	1	1	1	0	0	5-14
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	5-4	CPE	Call on parity even	1	1	1	0	1	1	0	0	5-14	
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	5-5	CPO	Call on parity odd	1	1	1	0	0	1	0	0	5-14	
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	5-5	RETURN											
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	5-5	RET	Return	1	1	0	0	1	0	0	1	5-14	
STAX B	Store A indirect	0	0	0	0	0	0	1	0	5-6	RC	Return on carry	1	1	0	1	1	0	0	0	5-14	
STAX D	Store A indirect	0	0	0	1	0	0	1	0	5-6	RNC	Return on no carry	1	1	0	1	0	0	0	0	5-14	
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	5-5	RZ	Return on zero	1	1	0	0	1	0	0	0	5-14	
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	5-5	RNZ	Return on no zero	1	1	0	0	0	0	0	0	5-14	
STA	Store A direct	0	0	1	1	0	0	1	0	5-5	RP	Return on positive	1	1	1	1	0	0	0	0	5-14	
LDA	Load A direct	0	0	1	1	1	0	1	0	5-5	RM	Return on minus	1	1	1	1	1	0	0	0	5-14	
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	5-5	RPE	Return on parity even	1	1	1	0	1	0	0	0	5-14	
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	5-5	RPO	Return on parity odd	1	1	1	0	0	0	0	0	5-14	
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	5-6	RESTART											
STACK OPS																						
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	5-15	RST	Restart	1	1	A	A	A	1	1	1	5-14	
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	5-15	INPUT/OUTPUT											
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	5-15	IN	Input	1	1	0	1	1	0	1	1	5-16	
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	5-15	OUT	Output	1	1	0	1	0	0	1	1	5-16	
POP B	Pop register Pair B & C off stack	1	1	0	0	0	0	0	1	5-15	INCREMENT AND DECREMENT											
POP D	Pop register Pair D & E off stack	1	1	0	1	0	0	0	1	5-15	INR r	Increment register	0	0	D	D	D	1	0	0	5-8	
POP H	Pop register Pair H & L off stack	1	1	1	0	0	0	0	1	5-15	DCR r	Decrement register	0	0	D	D	D	1	0	1	5-8	
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	5-15	INR M	Increment memory	0	0	1	1	0	1	0	0	5-8	
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	5-16	DCR M	Decrement memory	0	0	1	1	0	1	0	1	5-8	
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5-16	INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5-9	
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	5-5	INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5-9	
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5-9	INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5-9	
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5-9	DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5-9	
JUMP																						
JMP	Jump unconditional	1	1	0	0	0	0	1	1	5-13	DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5-9	
JC	Jump on carry	1	1	0	1	1	0	1	0	5-13	DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5-9	
JNC	Jump on no carry	1	1	0	1	0	0	1	0	5-13	DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	5-9	
JZ	Jump on zero	1	1	0	0	1	0	1	0	5-13	SUBTRACT											
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	5-13	SUB r	Subtract register from A	1	0	0	1	0	S	S	S	5-7	
JP	Jump on positive	1	1	1	1	0	0	1	0	5-13	SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	5-7	
JM	Jump on minus	1	1	1	1	1	0	1	0	5-13	SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	5-7	
JPE	Jump on parity even	1	1	1	0	1	0	1	0	5-13	SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	5-8	
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	5-13	SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	5-7	
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5-15												
CALL																						
CALL	Call unconditional	1	1	0	0	1	1	0	1	5-13												
CC	Call on carry	1	1	0	1	1	1	0	0	5-14												
CNC	Call on no carry	1	1	0	1	0	1	0	0	5-14												



## 8085A INSTRUCTION SET SUMMARY (Cont'd)

Table 5-3

		Instruction Code (1)									
Mnemonic	Description	D7	D6	D5	D4	D3	D2	D1	D0	Page	
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	5-8	
<b>LOGICAL</b>											
ANA r	And register with A	1	0	1	0	0	S	S	S	5-9	
XRA r	Exclusive OR register with A	1	0	1	0	1	S	S	S	5-10	
ORA r	OR register with A	1	0	1	1	0	S	S	S	5-10	
CMP r	Compare register with A	1	0	1	1	1	S	S	S	5-11	
ANA M	And memory with A	1	0	1	0	0	1	1	0	5-10	
XRA M	Exclusive OR memory with A	1	0	1	0	1	1	1	0	5-10	
ORA M	OR memory with A	1	0	1	1	0	1	1	0	5-11	
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	5-11	
ANI	And immediate with A	1	1	1	0	0	1	1	0	5-10	
XRI	Exclusive OR immediate with A	1	1	1	0	1	1	1	0	5-10	
ORI	OR immediate with A	1	1	1	1	0	1	1	0	5-11	
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	5-11	
<b>ROTATE</b>											
RLC	Rotate A left	0	0	0	0	0	1	1	1	5-11	

NOTES: 1. DDS or SSS: B 000, C 001, D 010, E 011, H 100, L 101, Memory 110, A 111.

2. Two possible cycle times. (6/12) indicate instruction cycles dependent on condition flags.

\*All mnemonics copyrighted © Intel Corporation 1976.

		Instruction Code (1)									
Mnemonic	Description	D7	D6	D5	D4	D3	D2	D1	D0	Page	
RRC	Rotate A right	0	0	0	0	1	1	1	1	5-12	
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	5-12	
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	5-12	
<b>SPECIALS</b>											
CMA	Complement A	0	0	1	0	1	1	1	1	5-12	
STC	Set carry	0	0	1	1	0	1	1	1	5-12	
CMC	Complement carry	0	0	1	1	1	1	1	1	5-12	
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	5-9	
<b>CONTROL</b>											
EI	Enable Interrupts	1	1	1	1	0	1	1	1	5-17	
DI	Disable Interrupt	1	1	1	1	0	0	1	1	5-17	
NOP	No-operation	0	0	0	0	0	0	0	0	5-17	
HLT	Halt	0	1	1	1	0	1	1	0	5-17	
<b>NEW 8085A INSTRUCTIONS</b>											
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0	5-17	
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0	5-18	

## Chapter 6

**MCS-85™**

**MCS-80™**

**Systems  
Support  
Components**

**Peripherals**

**Static RAMs**

**ROMs-EPROMs**

**MOS**

**6000**

**MOS**

**6000**



# SINGLE CHIP 8-BIT N-CANNEL MICROPROCESSORS

- **Single +5V Power Supply**
- **100% Software Compatible with 8080A**
- **1.3  $\mu$ s Instruction Cycle (8085A); 0.8  $\mu$ s (8085A-2)**
- **On-Chip Clock Generator (with External Crystal, LC or RC Network)**
- **On-Chip System Controller; Advanced Cycle Status Information Available for Large System Control**
- **Four Vectored Interrupt Inputs (One is non-Maskable) Plus an 8080A-compatible interrupt**
- **Serial In/Serial Out Port**
- **Decimal, Binary and Double Precision Arithmetic**
- **Direct Addressing Capability to 64k Bytes of Memory**

The Intel® 8085A is a complete 8 bit parallel Central Processing Unit (CPU). Its instruction set is 100% software compatible with the 8080A microprocessor, and it is designed to improve the present 8080A's performance by higher system speed. Its high level of system integration allows a minimum system of three IC's [8085A (CPU), 8156 (RAM/IO) and 8355/8755A (ROM/PROM/IO)] while maintaining total system expandability. The 8085A-2 is a faster version of the 8085A.

The 8085A incorporates all of the features that the 8224 (clock generator) and 8228 (system controller) provided for the 8080A, thereby offering a high level of system integration.

The 8085A uses a multiplexed data bus. The address is split between the 8 bit address bus and the 8 bit data bus. The on-chip address latches of 8155/8156/8355/8755A memory products allow a direct interface with the 8085A.

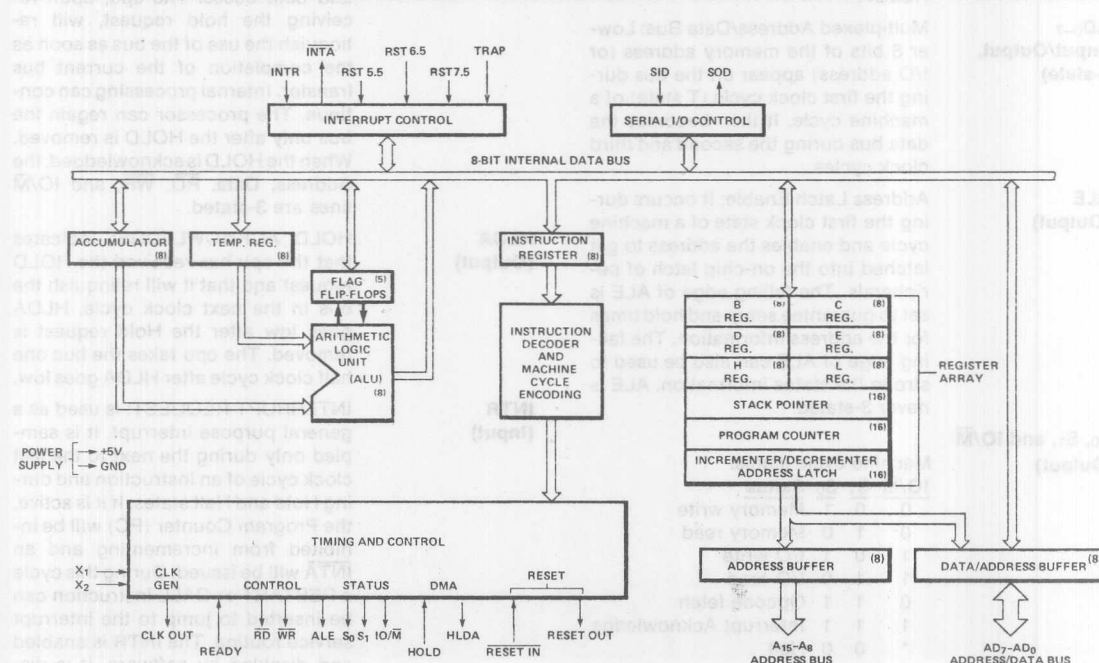


Figure 1. 8085A CPU Functional Block Diagram

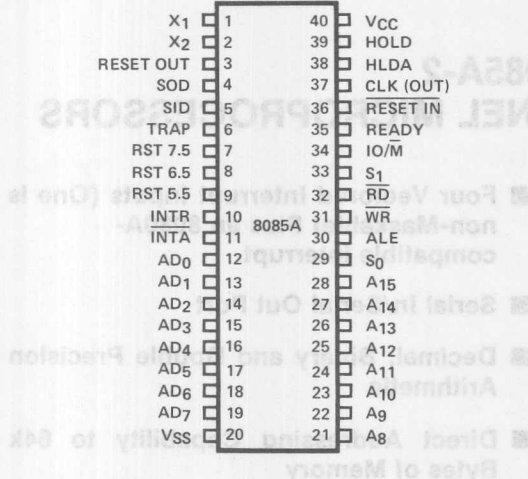


Figure 2. 8085A Pinout Diagram

## 8085A FUNCTIONAL PIN DEFINITION

The following describes the function of each pin:

### Symbol

### Function

**A<sub>8</sub>–A<sub>15</sub>**

**(Output, 3-state)**

**Address Bus:** The most significant 8 bits of the memory address or the 8 bits of the I/O address, 3-stated during Hold and Halt modes and during RESET.

**AD<sub>0</sub>–7**

**(Input/Output, 3-state)**

**Multiplexed Address/Data Bus:** Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state) of a machine cycle. It then becomes the data bus during the second and third clock cycles.

**ALE**

**(Output)**

**Address Latch Enable:** It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated.

**S<sub>0</sub>, S<sub>1</sub>, and IO/M**

**(Output)**

**Machine cycle status:**

IO/M	S <sub>1</sub>	S <sub>0</sub>	Status
0	0	1	Memory write
0	1	0	Memory read
1	0	1	I/O write
1	1	0	I/O read
0	1	1	Opcode fetch
1	1	1	Interrupt Acknowledge
*	0	0	Halt
*	X	X	Hold
*	X	X	Reset

\* = 3-state (high impedance)

X = unspecified

### Symbol

### Function

S<sub>1</sub> can be used as an advanced R/W status. IO/M, S<sub>0</sub> and S<sub>1</sub> become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines.

**RD**

**(Output, 3-state)**

**READ control:** A low level on RD indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer, 3-stated during Hold and Halt modes and during RESET.

**WR**

**(Output, 3-state)**

**WRITE control:** A low level on WR indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of WR. 3-stated during Hold and Halt modes and during RESET.

**READY**

**(Input)**

If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the cpu will wait an integral number of clock cycles for READY to go high before completing the read or write cycle.

**HOLD**

**(Input)**

**HOLD indicates** that another master is requesting the use of the address and data buses. The cpu, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3-stated.

**HLDA**

**(Output)**

**HOLD ACKNOWLEDGE:** Indicates that the cpu has received the HOLD request and that it will relinquish the bus in the next clock cycle. HLDA goes low after the Hold request is removed. The cpu takes the bus one half clock cycle after HLDA goes low.

**INTR**

**(Input)**

**INTERRUPT REQUEST:** is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.



## 8085A FUNCTIONAL PIN DESCRIPTION (Continued)

Symbol	Function
<b>INTA</b> (Output)	<b>INTERRUPT ACKNOWLEDGE:</b> Is used instead of (and has the same timing as) RD during the instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.
<b>RST 5.5</b> <b>RST 6.5</b> <b>RST 7.5</b> (Inputs)	<b>RESTART INTERRUPTS:</b> These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. The priority of these interrupts is ordered as shown in Table 1. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction.
<b>TRAP</b> (Input)	Trap interrupt is a nonmaskable RESTART interrupt. It is recognized at the same time as INTR or RST 5.5-7.5. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt. (See Table 1.)
<b>RESET IN</b> (Input)	Sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET with unpredictable results. RESET IN is a

Symbol	Function
<b>RESET OUT</b> (Output)	Indicates cpu is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods.
<b>X<sub>1</sub>, X<sub>2</sub></b> (Input)	X <sub>1</sub> and X <sub>2</sub> are connected to a crystal, LC, or RC network to drive the internal clock generator. X <sub>1</sub> can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency.
<b>CLK</b> (Output)	Clock Output for use as a system clock. The period of CLK is twice the X <sub>1</sub> , X <sub>2</sub> input period.
<b>SID</b> (Input)	Serial input data line. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
<b>SOD</b> (Output)	Serial output data line. The output SOD is set or reset as specified by the SIM instruction.
<b>V<sub>CC</sub></b>	+5 volt supply.
<b>V<sub>SS</sub></b>	Ground Reference.

TABLE 1. INTERRUPT PRIORITY, RESTART ADDRESS, AND SENSITIVITY

Name	Priority	Address Branched To (1) When Interrupt Occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled.
RST 7.5	2	3CH	Rising edge (latched).
RST 6.5	3	34H	High level until sampled.
RST 5.5	4	2CH	High level until sampled.
INTR	5	See Note (2).	High level until sampled.

## NOTES:

- (1) The processor pushes the PC on the stack before branching to the indicated address.  
 (2) The address branched to depends on the instruction provided to the cpu when the interrupt is acknowledged.

## FUNCTIONAL DESCRIPTION

The 8085A is a complete 8-bit parallel central processor. It is designed with N-channel depletion loads and requires a single +5 volt supply. Its basic clock speed is 3 MHz (8085A) or 5 MHz (8085A-2), thus improving on the present 8080A's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The cpu (8085A), a RAM/IO (8156), and a ROM or EPROM/IO chip (8355 or 8755A).

The 8085A has twelve addressable 8-bit registers. Four of them can function only as two 16-bit register pairs. Six others can be used interchangeably as 8-bit registers or as 16-bit register pairs. The 8085A register set is as follows:

Mnemonic	Register	Contents
ACC or A	Accumulator	8 bits
PC	Program Counter	16-bit address
BC,DE,HL	General-Purpose Registers; data pointer (HL)	8 bits x 6 or 16 bits x 3
SP	Stack Pointer	16-bit address
Flags or F	Flag Register	5 flags (8-bitspace)

The 8085A uses a multiplexed Data Bus. The address is split between the higher 8-bit Address Bus and the lower 8-bit Address/Data Bus. During the first T state (clock cycle) of a machine cycle the low order address is sent out on the Address/Data bus. These lower 8 bits may be latched externally by the Address Latch Enable signal (ALE). During the rest of the machine cycle the data bus is used for memory or I/O data.

The 8085A provides  $\overline{RD}$ ,  $\overline{WR}$ ,  $S_0$ ,  $S_1$ , and  $IO/\overline{M}$  signals for bus control. An Interrupt Acknowledge signal (INTA) is also provided. HOLD and all Interrupts are synchronized with the processor's internal clock. The 8085A also provides Serial Input Data (SID) and Serial Output Data (SOD) lines for simple serial interface.

In addition to these features, the 8085A has three maskable, vector interrupt pins and one nonmaskable TRAP interrupt.

## INTERRUPT AND SERIAL I/O

The 8085A has 5 interrupt inputs: INTR, RST 5.5, RST 6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080A INT. Each of the three RESTART inputs, 5.5, 6.5, and 7.5, has a programmable mask. TRAP is also a RESTART interrupt but it is nonmaskable.

The three maskable interrupts cause the internal execution of RESTART (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The non-maskable TRAP causes the internal execution of a RESTART vector independent of the state of the interrupt enable or masks. (See Table 1.)

There are two different types of inputs in the restart interrupts. RST 5.5 and RST 6.5 are *high level-sensitive* like INTR (and INT on the 8080) and are recognized with the same timing as INTR. RST 7.5 is *rising edge-sensitive*.

For RST 7.5, only a pulse is required to set an internal flip-flop which generates the internal interrupt request. (See Section 5.2.7.) The RST 7.5 request flip-flop remains

set until the request is serviced. Then it is reset automatically. This flip-flop may also be reset by using the SIM instruction or by issuing a RESET IN to the 8085A. The RST 7.5 internal flip-flop will be set by a pulse on the RST 7.5 pin even when the RST 7.5 interrupt is masked out.

The status of the three RST interrupt masks can only be affected by the SIM instruction and RESET IN. (See SIM, Chapter 5.

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP — highest priority, RST 7.5, RST 6.5, RST 5.5, INTR — lowest priority. This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST 5.5 can interrupt an RST 7.5 routine if the interrupts are re-enabled before the end of the RST 7.5 routine.

The TRAP interrupt is useful for catastrophic events such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both *edge and level sensitive*. The TRAP input must go high and remain high until it is acknowledged. It will not be recognized again until it goes low, then high again. This avoids any false triggering due to noise or logic glitches. Figure 3 illustrates the TRAP interrupt request circuitry within the 8085A. Note that the servicing of any interrupt (TRAP, RST 7.5, RST 6.5, RST 5.5, INTR) disables all future interrupts (except TRAPs) until an EI instruction is executed.

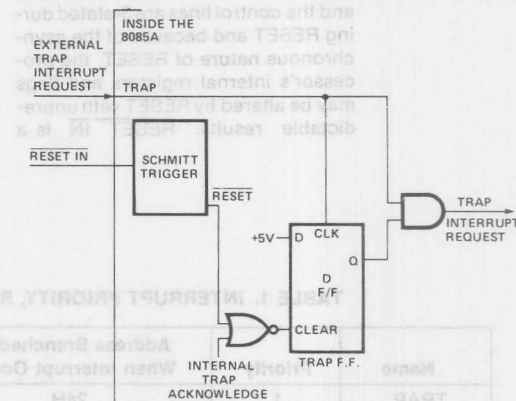


Figure 3. TRAP and RESET IN Circuit

The TRAP interrupt is special in that it disables interrupts, but preserves the previous interrupt enable status. Performing the first RIM instruction following a TRAP interrupt allows you to determine whether interrupts were enabled or disabled prior to the TRAP. All subsequent RIM instructions provide current interrupt enable status. Performing a RIM instruction following INTR, or RST 5.5–7.5 will provide current Interrupt Enable status, revealing that Interrupts are disabled. See the description of the RIM instruction in Chapter 5.

The serial I/O system is also controlled by the RIM and SIM instructions. SID is read by RIM, and SIM sets the SOD data.

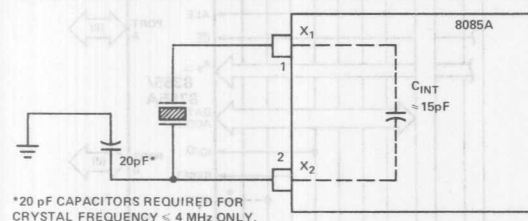
## DRIVING THE X<sub>1</sub> AND X<sub>2</sub> INPUTS

You may drive the clock inputs of the 8085A or 8085A-2 with a crystal, an LC tuned circuit, an RC network, or an external clock source. The driving frequency must be at least 1 MHz, and must be twice the desired internal clock frequency; hence, the 8085A is operated with a 6 MHz crystal (for 3 MHz clock), and the 8085A-2 can be operated with a 10 MHz crystal (for 5 MHz clock). If a crystal is used, it must have the following characteristics:

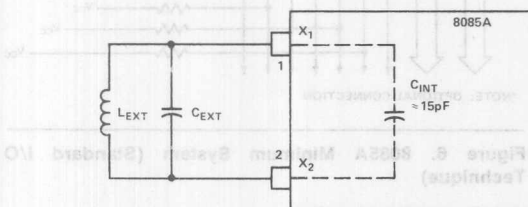
Parallel resonance at twice the clock frequency desired  
 $C_L$  (load capacitance)  $\leq 30$  pF  
 $C_s$  (shunt capacitance)  $\leq 7$  pF  
 $R_s$  (equivalent shunt resistance)  $\leq 75$  Ohms  
 Drive level: 10 mW  
 Frequency tolerance:  $\pm 0.05\%$  (suggested)

Note the use of the 20pF capacitor between X<sub>2</sub> and ground. This capacitor is required with crystal frequencies below 4 MHz to assure oscillator startup at the correct frequency. A parallel-resonant LC circuit may be used as the frequency-determining network for the 8085A, providing that its frequency tolerance of approximately  $\pm 10\%$  is acceptable. The components are chosen from the formula:

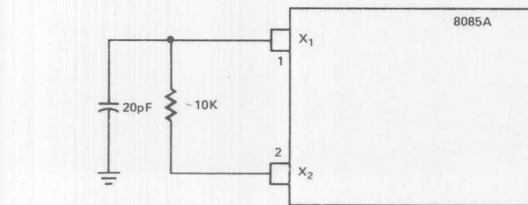
$$f = \frac{1}{2\pi\sqrt{L(C_{EXT} + C_{INT})}}$$



**A. Quartz Crystal Clock Driver**



**B. LC Tuned Circuit Clock Driver**



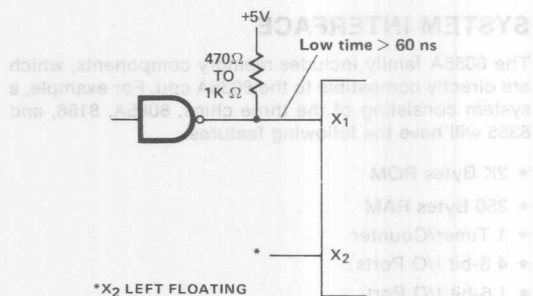
**C. RC Circuit Clock Driver**

To minimize variations in frequency, it is recommended that you choose a value for  $C_{EXT}$  that is at least twice that of  $C_{INT}$ , or 30 pF. The use of an LC circuit is not recommended for frequencies higher than approximately 5 MHz.

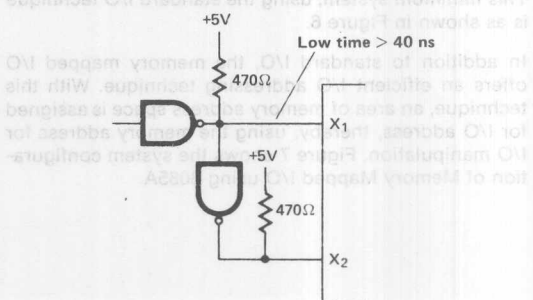
An RC circuit may be used as the frequency-determining network for the 8085A if maintaining a precise clock frequency is of no importance. Variations in the on-chip timing generation can cause a wide variation in frequency when using the RC mode. Its advantage is its low component cost. The driving frequency generated by the circuit shown is approximately 3 MHz. It is not recommended that frequencies greatly higher or lower than this be attempted.

Figure 4 shows the recommended clock driver circuits. Note in D and E that pullup resistors are required to assure that the high level voltage of the input is at least 4 V.

For driving frequencies up to and including 6 MHz you may supply the driving signal to X<sub>1</sub> and leave X<sub>2</sub> open-circuited (Figure 4D). If the driving frequency is from 6 MHz to 10 MHz, stability of the clock generator will be improved by driving both X<sub>1</sub> and X<sub>2</sub> with a push-pull source (Figure 4E). To prevent self-oscillation of the 8085A, be sure that X<sub>2</sub> is not coupled back to X<sub>1</sub> through the driving circuit.



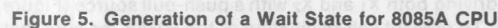
**D. 1-6 MHz Input Frequency External Clock Driver Circuit**



**E. 1-10 MHz Input Frequency External Clock Driver Circuit**

**Figure 4. Clock Driver Circuits**

- CLK is rising edge-triggered
- CLEAR is low-level active.



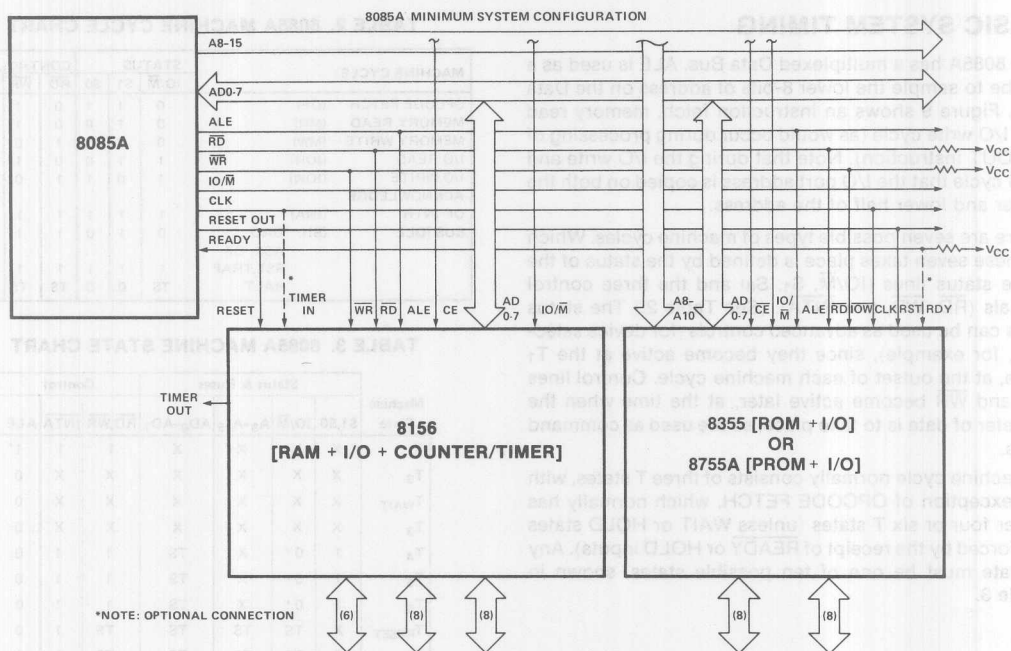
## SYSTEM INTERFACE

- 2K Bytes ROM
- 256 Bytes RAM
- 1 Timer/Counter
- 4 8-bit I/O Ports
- 1 6-bit I/O Port
- 4 Interrupt Levels
- Serial In/Serial Out Ports

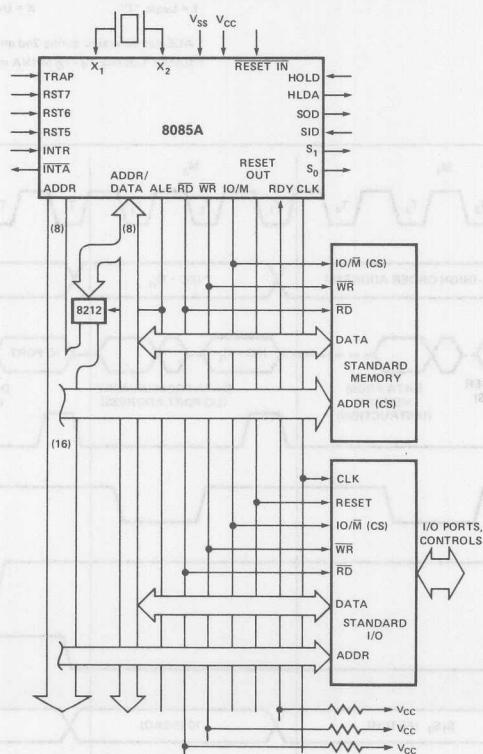
In addition to standard I/O, the memory mapped I/O offers an efficient I/O addressing technique. With this technique, an area of memory address space is assigned for I/O address, thereby, using the memory address for I/O manipulation. Figure 7 shows the system configuration of Memory Mapped I/O using 8085A.

[illegible]

**Figure 6. 8085A Minimum System (Standard I/O Technique)**



**Figure 7. MCS-85™ Minimum System (Memory Mapped I/O)**



**Figure 8. MCS-85™ System (Using Standard Memories)**



## BASIC SYSTEM TIMING

The 8085A has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8-bits of address on the Data Bus. Figure 9 shows an instruction fetch, memory read and I/O write cycle (as would occur during processing of the OUT instruction). Note that during the I/O write and read cycle that the I/O port address is copied on both the upper and lower half of the address.

There are seven possible types of machine cycles. Which of these seven takes place is defined by the status of the three status lines ( $\overline{IO/\overline{M}}$ ,  $S_1$ ,  $S_0$ ) and the three control signals ( $\overline{RD}$ ,  $\overline{WR}$ , and  $\overline{INTA}$ ). (See Table 2.) The status lines can be used as advanced controls (for device selection, for example), since they become active at the  $T_1$  state, at the outset of each machine cycle. Control lines  $\overline{RD}$  and  $\overline{WR}$  become active later, at the time when the transfer of data is to take place, so are used as command lines.

A machine cycle normally consists of three T states, with the exception of OPCODE FETCH, which normally has either four or six T states (unless WAIT or HOLD states are forced by the receipt of  $\overline{READY}$  or  $\overline{HOLD}$  inputs). Any T state must be one of ten possible states, shown in Table 3.

TABLE 2. 8085A MACHINE CYCLE CHART

MACHINE CYCLE		STATUS			CONTROL		
		$\overline{IO/\overline{M}}$	$S_1$	$S_0$	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
OPCODE FETCH (OF)		0	1	1	0	1	1
MEMORY READ (MR)		0	1	0	0	1	1
MEMORY WRITE (MW)		0	0	1	1	0	1
I/O READ (IOR)		1	1	0	0	1	1
I/O WRITE (IOW)		1	0	1	1	0	1
ACKNOWLEDGE							
OF INTR (INA)		1	1	1	1	1	0
BUS IDLE (BI): DAD		0	1	0	1	1	1
	ACK. OF RST, TRAP	1	1	1	1	1	1
	HALT	TS	0	0	TS	TS	1

TABLE 3. 8085A MACHINE STATE CHART

Machine State	Status & Buses				Control		
	$S_1, S_0$	$\overline{IO/\overline{M}}$	$A_8-A_{15}$	$AD_0-AD_7$	$\overline{RD}, \overline{WR}$	$\overline{INTA}$	ALE
$T_1$	X	X	X	X	1	1	1*
$T_2$	X	X	X	X	X	X	0
$T_{WAIT}$	X	X	X	X	X	X	0
$T_3$	X	X	X	X	X	X	0
$T_4$	1	0†	X	TS	1	1	0
$T_5$	1	0†	X	TS	1	1	0
$T_6$	1	0†	X	TS	1	1	0
$T_{RESET}$	X	TS	TS	TS	TS	1	0
$T_{HALT}$	0	TS	TS	TS	TS	1	0
$T_{HOLD}$	X	TS	TS	TS	TS	1	0

0 = Logic "0"

TS = High Impedance

1 = Logic "1"

X = Unspecified

\* ALE not generated during 2nd and 3rd machine cycles of DAD instruction.

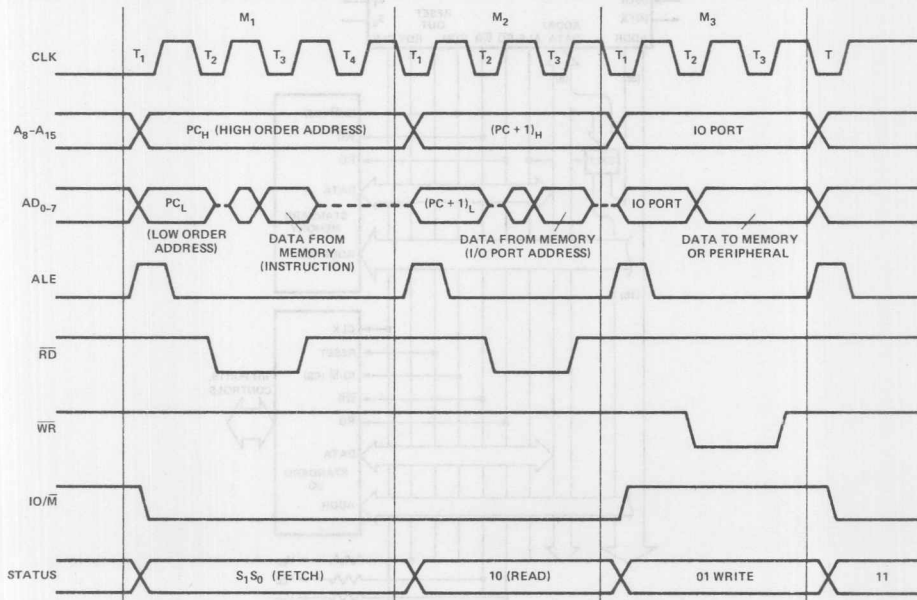
†  $\overline{IO/\overline{M}} = 1$  during  $T_4-T_6$  of INA machine cycle.

Figure 9. 8085A Basic System Timing

TABLE 4. ABSOLUTE MAXIMUM RATINGS\*

Ambient Temperature Under Bias . . . . .	0°C to 70°C
Storage Temperature . . . . .	-65°C to +150°C
Voltage on Any Pin . . . . .	
With Respect to Ground . . . . .	-0.5V to +7V
Power Dissipation . . . . .	1.5 Watt

## \*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

TABLE 5. D.C. CHARACTERISTICS

( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5\text{V} \pm 5\%$ ;  $V_{SS} = 0\text{V}$ ; unless otherwise specified)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
$V_{IL}$	Input Low Voltage	-0.5	+0.8	V	
$V_{IH}$	Input High Voltage	2.0	$V_{CC} + 0.5$	V	
$V_{OL}$	Output Low Voltage		0.45	V	$I_{OL} = 2\text{mA}$
$V_{OH}$	Output High Voltage	2.4		V	$I_{OH} = -400\mu\text{A}$
$I_{CC}$	Power Supply Current		170	mA	
$I_{IL}$	Input Leakage		$\pm 10$	$\mu\text{A}$	$V_{in} = V_{CC}$
$I_{LO}$	Output Leakage		$\pm 10$	$\mu\text{A}$	$0.45\text{V} \leq V_{out} \leq V_{CC}$
$V_{ILR}$	Input Low Level, RESET	-0.5	+0.8	V	
$V_{IHR}$	Input High Level, RESET	2.4	$V_{CC} + 0.5$	V	
$V_{HY}$	Hysteresis, RESET	0.25		V	

TABLE 6. A.C. CHARACTERISTICS

T<sub>A</sub> = 0°C to 70°C; V<sub>CC</sub> = 5V ± 5%; V<sub>SS</sub> = 0V

Symbol	Parameter	8085A <sup>[2]</sup>		8085A-2 <sup>[2]</sup> (Preliminary)		Units
		Min.	Max.	Min.	Max.	
t <sub>CYC</sub>	CLK Cycle Period	320	2000	200	2000	ns
t <sub>1</sub>	CLK Low Time (Standard CLK Loading)	80		40		ns
t <sub>2</sub>	CLK High Time (Standard CLK Loading)	120		70		ns
t <sub>r</sub> , t <sub>f</sub>	CLK Rise and Fall Time		30		30	ns
t <sub>XKR</sub>	X <sub>1</sub> Rising to CLK Rising	30	120	30	100	ns
t <sub>XKF</sub>	X <sub>1</sub> Rising to CLK Falling	30	150	30	110	ns
t <sub>AC</sub>	A <sub>8-15</sub> Valid to Leading Edge of Control <sup>[1]</sup>	270		115		ns
t <sub>ACL</sub>	A <sub>0-7</sub> Valid to Leading Edge of Control	240		115		ns
t <sub>AD</sub>	A <sub>0-15</sub> Valid to Valid Data In		575		350	ns
t <sub>AFR</sub>	Address Float After Leading Edge of READ (INTA)		0		0	ns
t <sub>AL</sub>	A <sub>8-15</sub> Valid Before Trailing Edge of ALE <sup>[1]</sup>	115		50		ns
t <sub>ALL</sub>	A <sub>0-7</sub> Valid Before Trailing Edge of ALE	90		50		ns
t <sub>ARY</sub>	READY Valid from Address Valid		220		100	ns
t <sub>CA</sub>	Address (A <sub>8-15</sub> ) Valid After Control	120		60		ns
t <sub>CC</sub>	Width of Control Low ( $\overline{RD}$ , $\overline{WR}$ , $\overline{INTA}$ ) Edge of ALE	400		230		ns
t <sub>CL</sub>	Trailing Edge of Control to Leading Edge of ALE	50		25		ns
t <sub>DW</sub>	Data Valid to Trailing Edge of $\overline{WRITE}$	420		230		ns
t <sub>HABE</sub>	HLDA to Bus Enable		210		150	ns
t <sub>HABF</sub>	Bus Float After HLDA		210		150	ns
t <sub>HACK</sub>	HLDA Valid to Trailing Edge of CLK	110		40		ns
t <sub>HDH</sub>	HOLD Hold Time	0		0		ns
t <sub>HDS</sub>	HOLD Setup Time to Trailing Edge of CLK	170		120		ns
t <sub>INH</sub>	INTR Hold Time	0		0		ns
t <sub>INS</sub>	INTR, RST, and TRAP Setup Time to Falling Edge of CLK	160		150		ns
t <sub>LA</sub>	Address Hold Time After ALE	100		50		ns
t <sub>LC</sub>	Trailing Edge of ALE to Leading Edge of Control	130		60		ns
t <sub>LCK</sub>	ALE Low During CLK High	100		50		ns
t <sub>LDR</sub>	ALE to Valid Data During Read		460		270	ns
t <sub>LDW</sub>	ALE to Valid Data During Write		200		120	ns
t <sub>LL</sub>	ALE Width	140		80		ns
t <sub>LRV</sub>	ALE to READY Stable		110		30	ns

Table 6. A.C. Characteristics (Cont.)

Symbol	Parameter	8085A <sup>[2]</sup>		8085A-2 <sup>[2]</sup> (Preliminary)		Units
		Min.	Max.	Min.	Max.	
$t_{RAE}$	Trailing Edge of $\overline{RD}$ to Re-Enabling of Address	150		90		ns
$t_{RD}$	$\overline{RD}$ (or $\overline{INTA}$ ) to Valid Data		300		150	ns
$t_{RV}$	Control Trailing Edge to Leading Edge of Next Control	400		220		ns
$t_{RDH}$	Data Hold Time After $\overline{RD}$ $\overline{INTA}$ <sup>[7]</sup>	0		0		ns
$t_{RYH}$	READY Hold Time	0		0		ns
$t_{RYS}$	READY Setup Time to Leading Edge of CLK	110		100		ns
$t_{WD}$	Data Valid After Trailing Edge of $\overline{WR}$	100		60		ns
$t_{WDL}$	LEADING Edge of $\overline{WR}$ to Data Valid		40		20	ns

## Notes:

- $A_8-A_{15}$  address Specs apply to  $\overline{IO/\overline{M}}$ ,  $S_0$ , and  $S_1$  except  $A_8-A_{15}$  are undefined during  $T_4-T_6$  of OF cycle whereas  $\overline{IO/\overline{M}}$ ,  $S_0$ , and  $S_1$  are stable.
- Test conditions:  $t_{CYC} = 320$  ns (8085A)/200 ns (8085A-2);  $C_L = 150$  pF.
- For all output timing where  $C_L = 150$  pF use the following correction factors:  
 $25 \text{ pF} \leq C_L < 150 \text{ pF}$ :  $-0.10 \text{ ns/pF}$   
 $150 \text{ pF} < C_L \leq 300 \text{ pF}$ :  $+0.30 \text{ ns/pF}$
- Output timings are measured with purely capacitive load.
- All timings are measured at output voltage  $V_L = 0.8\text{V}$ ,  $V_H = 2.0\text{V}$ , and  $1.5\text{V}$  with 20 ns rise and fall time on inputs.
- To calculate timing specifications at other values of  $t_{CYC}$  use Table 7.
- Data hold time is guaranteed under all loading conditions.

## Input Waveform for A.C. Tests:

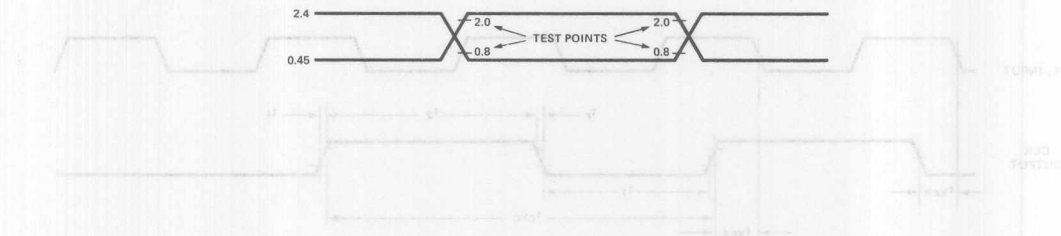


Figure 10. Clock Timing Waveform

TABLE 7. BUS TIMING SPECIFICATION AS A  $T_{CYC}$  DEPENDENT

8085A				8085A-2 (Preliminary)			
$t_{AL}$	—	$(1/2) T - 45$	MIN	$t_{AL}$	—	$(1/2) T - 50$	MIN
$t_{LA}$	—	$(1/2) T - 60$	MIN	$t_{LA}$	—	$(1/2) T - 50$	MIN
$t_{LL}$	—	$(1/2) T - 20$	MIN	$t_{LL}$	—	$(1/2) T - 20$	MIN
$t_{LCK}$	—	$(1/2) T - 60$	MIN	$t_{LCK}$	—	$(1/2) T - 50$	MIN
$t_{LC}$	—	$(1/2) T - 30$	MIN	$t_{LC}$	—	$(1/2) T - 40$	MIN
$t_{AD}$	—	$(5/2 + N) T - 225$	MAX	$t_{AD}$	—	$(5/2 + N) T - 150$	MAX
$t_{RD}$	—	$(3/2 + N) T - 180$	MAX	$t_{RD}$	—	$(3/2 + N) T - 150$	MAX
$t_{RAE}$	—	$(1/2) T - 10$	MIN	$t_{RAE}$	—	$(1/2) T - 10$	MIN
$t_{CA}$	—	$(1/2) T - 40$	MIN	$t_{CA}$	—	$(1/2) T - 40$	MIN
$t_{DW}$	—	$(3/2 + N) T - 60$	MIN	$t_{DW}$	—	$(3/2 + N) T - 70$	MIN
$t_{WD}$	—	$(1/2) T - 60$	MIN	$t_{WD}$	—	$(1/2) T - 40$	MIN
$t_{CC}$	—	$(3/2 + N) T - 80$	MIN	$t_{CC}$	—	$(3/2 + N) T - 70$	MIN
$t_{CL}$	—	$(1/2) T - 110$	MIN	$t_{CL}$	—	$(1/2) T - 75$	MIN
$t_{ARY}$	—	$(3/2) T - 260$	MAX	$t_{ARY}$	—	$(3/2) T - 200$	MAX
$t_{HACK}$	—	$(1/2) T - 50$	MIN	$t_{HACK}$	—	$(1/2) T - 60$	MIN
$t_{HABF}$	—	$(1/2) T + 50$	MAX	$t_{HABF}$	—	$(1/2) T + 50$	MAX
$t_{HABE}$	—	$(1/2) T + 50$	MAX	$t_{HABE}$	—	$(1/2) T + 50$	MAX
$t_{AC}$	—	$(2/2) T - 50$	MIN	$t_{AC}$	—	$(2/2) T - 85$	MIN
$t_1$	—	$(1/2) T - 80$	MIN	$t_1$	—	$(1/2) T - 60$	MIN
$t_2$	—	$(1/2) T - 40$	MIN	$t_2$	—	$(1/2) T - 30$	MIN
$t_{RV}$	—	$(3/2) T - 80$	MIN	$t_{RV}$	—	$(3/2) T - 80$	MIN
$t_{LDR}$	—	$(4/2) T - 180$	MAX	$t_{LDR}$	—	$(4/2) T - 130$	MAX

NOTE: N is equal to the total WAIT states.

$$T = t_{CYC}$$

NOTE: N is equal to the total WAIT states.

$$T = t_{CYC}$$

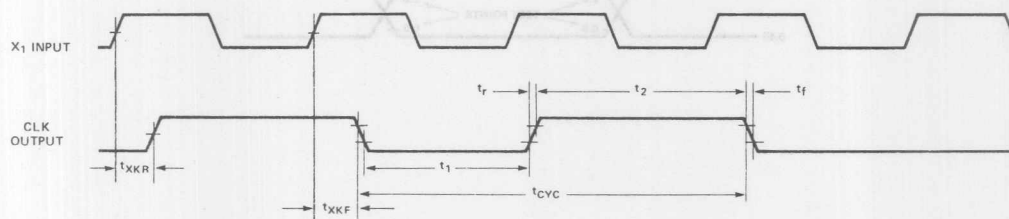
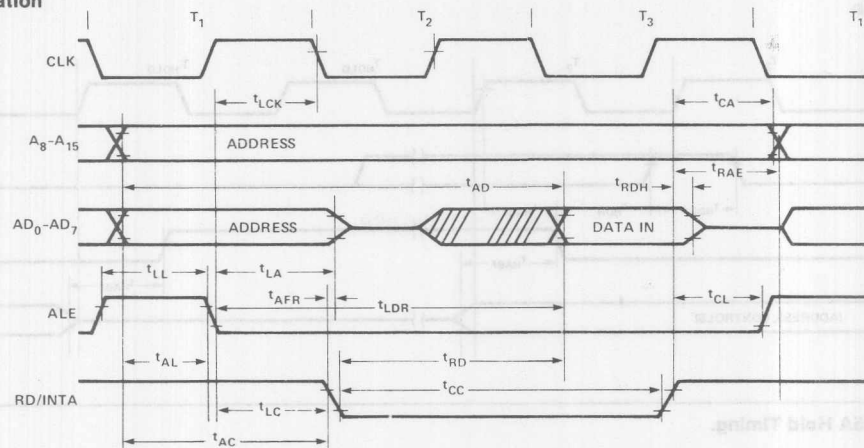


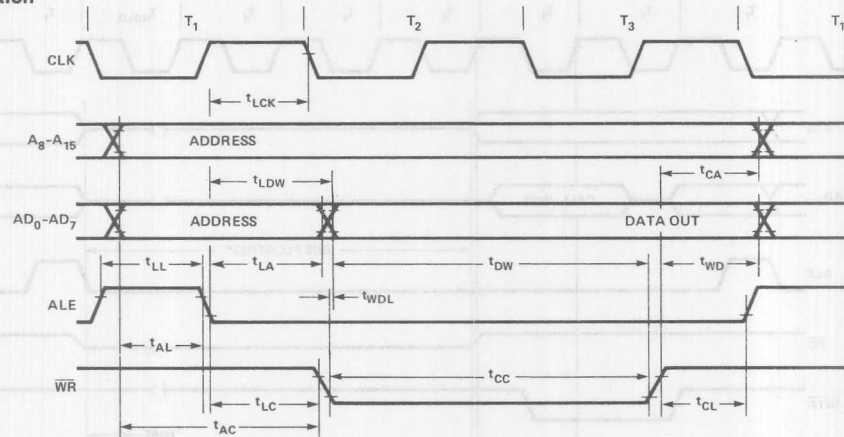
Figure 10. Clock Timing Waveform



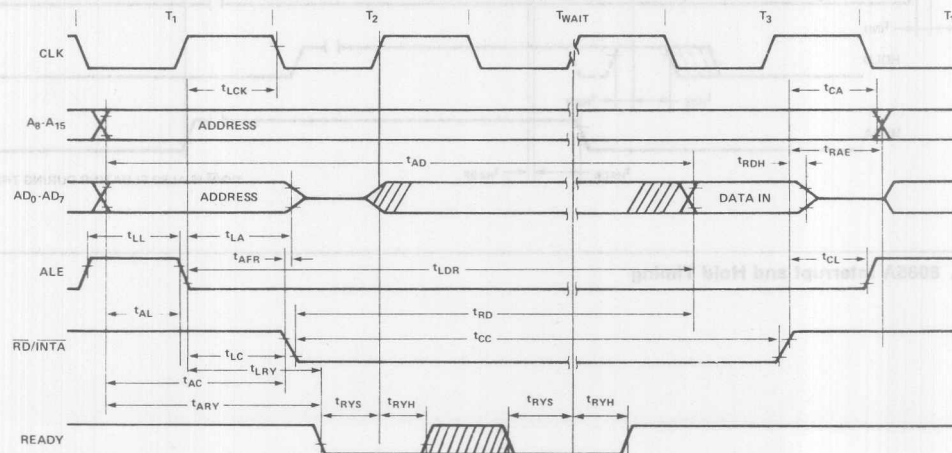
## Read Operation



## Write Operation



## Read operation with Wait Cycle (Typical) — same READY timing applies to WRITE operation.



NOTE 1: READY MUST REMAIN STABLE DURING SETUP AND HOLD TIMES.

Figure 11. 8085A Bus Timing, With and Without Wait

## Hold Operation

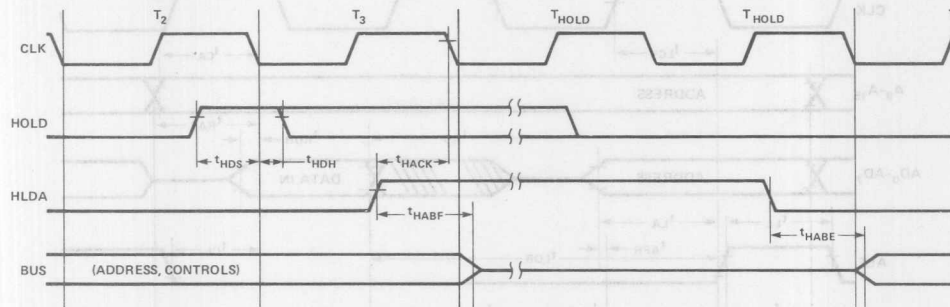


Figure 12. 8085A Hold Timing.

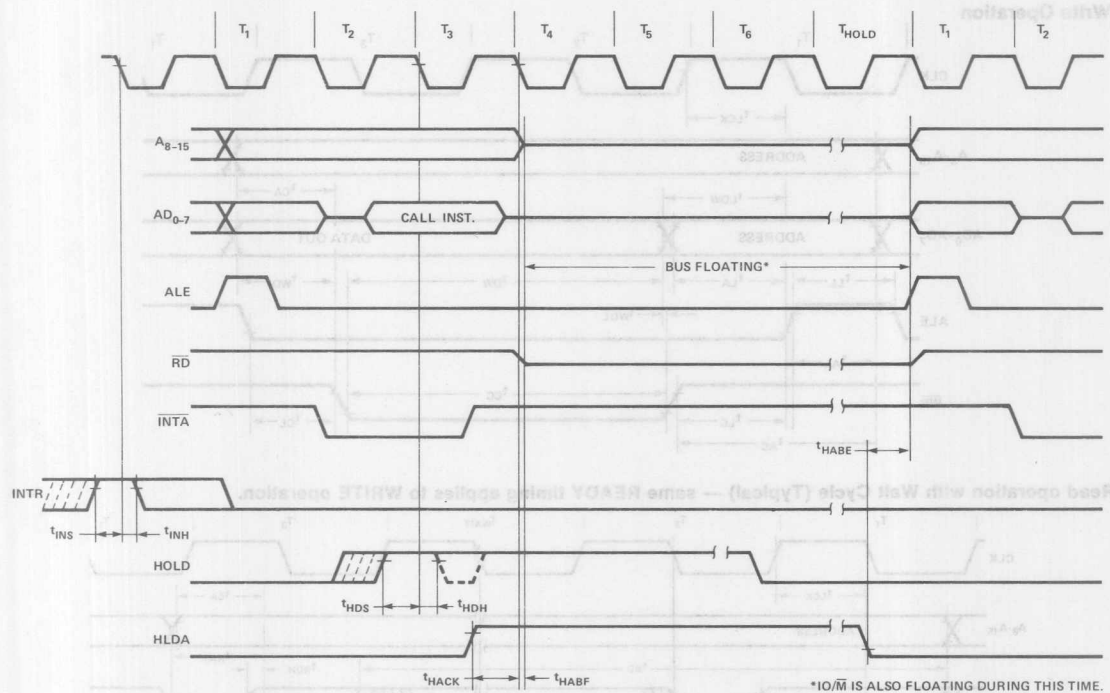


Figure 13. 8085A Interrupt and Hold Timing

## 8085A INSTRUCTION SET SUMMARY BY FUNCTIONAL GROUPING

Table 6-1

		Instruction Code (1)								Instruction Code (1)												
Mnemonic	Description	D7	D6	D5	D4	D3	D2	D1	D0	Page	Mnemonic	Description	D7	D6	D5	D4	D3	D2	D1	D0	Page	
MOVE, LOAD, AND STORE																						
MOV r1 r2	Move register to register	0	1	0	0	0	S	S	S	5-4	CZ	Call on zero	1	1	0	0	1	1	0	0	5-14	
MOV M.r	Move register to memory	0	1	1	1	0	S	S	S	5-4	CNZ	Call on no zero	1	1	0	0	0	1	0	0	5-14	
MOV r.M	Move memory to register	0	1	0	0	0	1	1	0	5-4	CP	Call on positive	1	1	1	1	0	1	0	0	5-14	
MVI r	Move immediate register	0	0	0	0	0	1	1	0	5-4	CM	Call on minus	1	1	1	1	1	1	0	0	5-14	
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	5-4	CPE	Call on parity even	1	1	1	0	1	1	0	0	5-14	
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	5-5	CPO	Call on parity odd	1	1	1	0	0	1	0	0	5-14	
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	5-5	RETURN											
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	5-5	RET	Return	1	1	0	0	1	0	0	1	5-14	
STAX B	Store A indirect	0	0	0	0	0	0	1	0	5-6	RC	Return on carry	1	1	0	1	1	0	0	0	5-14	
STAX D	Store A indirect	0	0	0	1	0	0	1	0	5-6	RNC	Return on no carry	1	1	0	1	0	0	0	0	5-14	
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	5-5	RZ	Return on zero	1	1	0	0	1	0	0	0	5-14	
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	5-5	RNZ	Return on no zero	1	1	0	0	0	0	0	0	5-14	
STA	Store A direct	0	0	1	1	0	0	1	0	5-5	RP	Return on positive	1	1	1	1	0	0	0	0	5-14	
LDA	Load A direct	0	0	1	1	1	0	1	0	5-5	RM	Return on minus	1	1	1	1	1	0	0	0	5-14	
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	5-5	RPE	Return on parity even	1	1	1	0	1	0	0	0	5-14	
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	5-5	RPO	Return on parity odd	1	1	1	0	0	0	0	0	5-14	
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	5-6	RESTART											
STACK OPS																						
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	5-15	RST	Restart	1	1	A	A	A	1	1	1	5-14	
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	5-15	INPUT/OUTPUT											
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	5-15	IN	Input	1	1	0	1	1	0	1	1	5-16	
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	5-15	OUT	Output	1	1	0	1	0	0	1	1	5-16	
POP B	Pop register Pair B & C off stack	1	1	0	0	0	0	0	1	5-15	INCREMENT AND DECREMENT											
POP D	Pop register Pair D & E off stack	1	1	0	1	0	0	0	1	5-15	INR r	Increment register	0	0	0	0	0	1	0	0	5-8	
POP H	Pop register Pair H & L off stack	1	1	1	0	0	0	0	1	5-15	DCR r	Decrement register	0	0	0	0	0	1	0	1	5-8	
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	5-15	INR M	Increment memory	0	0	1	1	0	1	0	0	5-8	
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	5-16	DCR M	Decrement memory	0	0	1	1	0	1	0	1	5-8	
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5-16	INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5-9	
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	5-5	INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5-9	
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5-9	INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5-9	
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5-9	DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5-9	
JUMP																						
JMP	Jump unconditional	1	1	0	0	0	0	1	1	5-13	DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5-9	
JC	Jump on carry	1	1	0	1	1	0	1	0	5-13	DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5-9	
JNC	Jump on no carry	1	1	0	1	0	0	1	0	5-13	ADD											
JZ	Jump on zero	1	1	0	0	1	0	1	0	5-13	ADD r	Add register to A	1	0	0	0	0	S	S	S	5-6	
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	5-13	ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	5-6	
JP	Jump on positive	1	1	1	1	0	0	1	0	5-13	ADD M	Add memory to A	1	0	C	0	0	1	1	0	5-6	
JM	Jump on minus	1	1	1	1	1	0	1	0	5-13	ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	5-7	
JPE	Jump on parity even	1	1	1	0	1	0	1	0	5-13	ADI	Add immediate to A	1	1	0	0	0	1	1	0	5-6	
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	5-13	ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	5-7	
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5-15	DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	5-9	
CALL																						
CALL	Call unconditional	1	1	0	0	1	1	0	1	5-13	DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	5-9	
CC	Call on carry	1	1	0	1	1	1	0	0	5-14	DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	5-9	
CNC	Call on no carry	1	1	0	1	0	1	0	0	5-14	DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	5-9	
SUBTRACT																						
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	5-7	SUI											
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	5-7	SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	5-7	
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	5-7												
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	1	0	5-8											

## 8085A INSTRUCTION SET SUMMARY (Cont'd)

Table 6-1

Mnemonic	Description	Instruction Code (1)								Page
		D7	D6	D5	D4	D3	D2	D1	D0	
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	5-8
<b>LOGICAL</b>										
ANA r	And register with A	1	0	1	0	0	S	S	S	5-9
XRA r	Exclusive OR register with A	1	0	1	0	1	S	S	S	5-10
ORA r	OR register with A	1	0	1	1	0	S	S	S	5-10
CMP r	Compare register with A	1	0	1	1	1	S	S	S	5-11
ANA M	And memory with A	1	0	1	0	0	1	1	0	5-10
XRA M	Exclusive OR memory with A	1	0	1	0	1	1	1	0	5-10
ORA M	OR memory with A	1	0	1	1	0	1	1	0	5-11
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	5-11
ANI	And immediate with A	1	1	1	0	0	1	1	0	5-10
XRI	Exclusive OR immediate with A	1	1	1	0	1	1	1	0	5-10
ORI	OR immediate with A	1	1	1	1	0	1	1	0	5-11
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	5-11
<b>ROTATE</b>										
RLC	Rotate A left	0	0	0	0	0	1	1	1	5-11
<b>NEW 8085A INSTRUCTIONS</b>										
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0	5-17
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0	5-18

NOTES: 1. DDS or SSS: B 000, C 001, D 010, E 011, H 100, L 101, Memory 110, A 111.

2. Two possible cycle times. (6/12) indicate instruction cycles dependent on condition flags.

\*All mnemonics copyrighted © Intel Corporation 1976.

# 8155/8156/8155-2/8156-2

## 2048 BIT STATIC MOS RAM WITH I/O PORTS AND TIMER

- 256 Word x 8 Bits
- Single +5V Power Supply
- Completely Static Operation
- Internal Address Latch
- 2 Programmable 8 Bit I/O Ports

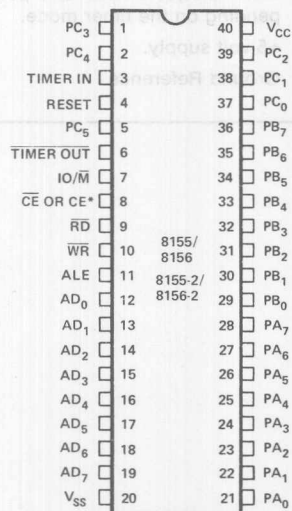
- 1 Programmable 6-Bit I/O Port
- Programmable 14-Bit Binary Counter/Timer
- Compatible with 8085A and 8088 CPU
- Multiplexed Address and Data Bus
- 40 Pin DIP

The 8155 and 89156 are RAM and I/O chips to be used in the 8085A and 8088 microprocessor systems. The RAM portion is designed with 2048 static cells organized as 256 x 8. They have a maximum access time of 400 ns to permit use with no wait states in 8085A CPU. The 8155-2 and 8156-2 have maximum access times of 330 ns for use with the 8085A-2 and the full speed 5 MHz 8088 CPU.

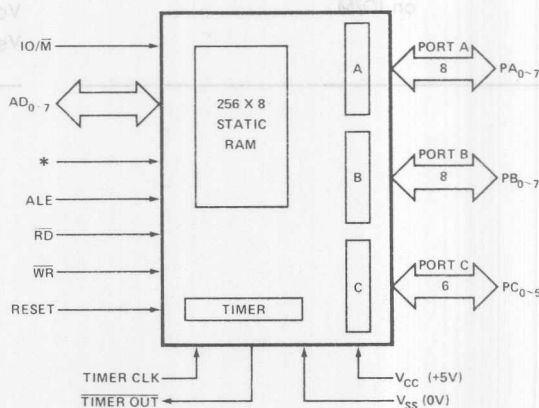
The I/O portion consists of three general purpose I/O ports. One of the three ports can be programmed to be status pins, thus allowing the other two ports to operate in handshake mode.

A 14-bit programmable counter/timer is also included on chip to provide either a square wave or terminal count pulse for the CPU system depending on timer mode.

### PIN CONFIGURATION



### BLOCK DIAGRAM

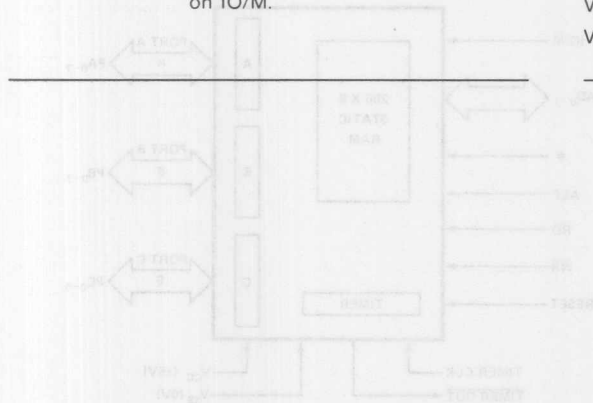


\*: 8155/8155-2 =  $\overline{\text{CE}}$ , 8156/8156-2 = CE



## 8155/8156 PIN FUNCTIONS

Symbol	Function	Symbol	Function
RESET (input)	Pulse provided by the 8085A to initialize the system (connect to 8085A RESET OUT). Input high on this line resets the chip and initializes the three I/O ports to input mode. The width of RESET pulses should typically be two 8085A clock cycle times.	ALE (input)	Address Latch Enable: This control signal latches both the address on the AD <sub>0-7</sub> lines and the state of the Chip Enable and IO/ $\overline{M}$ into the chip at the falling edge of ALE.
AD <sub>0-7</sub> (input)	3-state Address/Data lines that interface with the CPU lower 8-bit Address/Data Bus. The 8-bit address is latched into the address latch inside the 8155/56 on the falling edge of ALE. The address can be either for the memory section or the I/O section depending on the IO/ $\overline{M}$ input. The 8-bit data is either written into the chip or read from the chip, depending on the $\overline{WR}$ or $\overline{RD}$ input signal.	IO/ $\overline{M}$ (input)	Selects memory if low and I/O and command/status registers if high.
CE or $\overline{CE}$ (input)	Chip Enable: On the 8155, this pin is $\overline{CE}$ and is ACTIVE LOW. On the 8156, this pin is CE and is ACTIVE HIGH.	PA <sub>0-7</sub> (8) (input/output)	These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register.
$\overline{RD}$ (input)	Read control: Input low on this line with the Chip Enable active enables and AD <sub>0-7</sub> buffers. If IO/ $\overline{M}$ pin is low, the RAM content will be read out to the AD bus. Otherwise the content of the selected I/O port or command/status registers will be read to the AD bus.	PB <sub>0-7</sub> (8) (input/output)	These 8 pins are general purpose I/O pins. The in/out direction is selected by programming the command register.
$\overline{WR}$ (input)	Write control: Input low on this line with the Chip Enable active causes the data on the Address/Data bus to be written to the RAM or I/O ports and command/status register depending on IO/ $\overline{M}$ .	PC <sub>0-5</sub> (6) (input/output)	These 6 pins can function as either input port, output port, or as control signals for PA and PB. Programming is done through the command register. When PC <sub>0-5</sub> are used as control signals, they will provide the following: PC <sub>0</sub> — A INTR (Port A Interrupt) PC <sub>1</sub> — A BF (Port A Buffer Full) PC <sub>2</sub> — A STB (Port A Strobe) PC <sub>3</sub> — B INTR (Port B Interrupt) PC <sub>4</sub> — B BF (Port B Buffer Full) PC <sub>5</sub> — B STB (Port B Strobe)
		TIMER IN (input)	Input to the counter-timer.
		TIMER OUT (output)	Timer output. This output can be either a square wave or a pulse depending on the timer mode.
		VCC	+5 volt supply.
		VSS	Ground Reference.



## DESCRIPTION

The 8155/8156 contains the following:

- 2k Bit Static RAM organized as 256 x 8
- Two 8-bit I/O ports (PA & PB) and one 6-bit I/O port (PC)
- 14-bit timer-counter

The  $\text{IO}/\overline{\text{M}}$  (IO/Memory Select) pin selects either the five registers (Command, Status, PA0-7, PB0-7, PC0-5) or the memory (RAM) portion. (See Figure 1.)

The 8-bit address on the Address/Data lines, Chip Enable input CE or  $\overline{\text{CE}}$ , and  $\text{IO}/\overline{\text{M}}$  are all latched on-chip at the falling edge of ALE. (See Figure 2.)

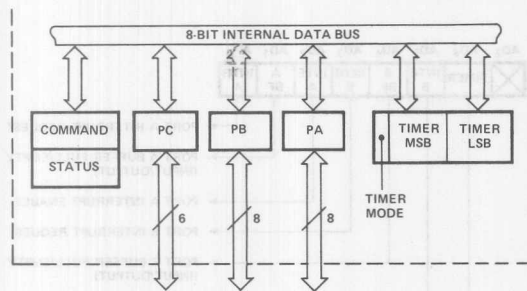
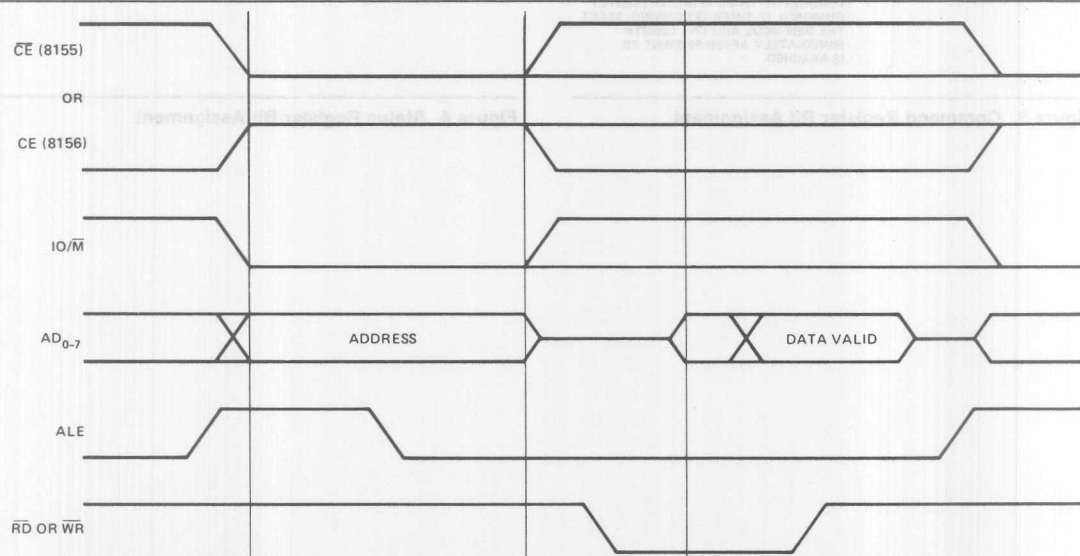


Figure 1. 8155/8156 Internal Registers



NOTE: FOR DETAILED TIMING INFORMATION, SEE FIGURE 12 AND A.C. CHARACTERISTICS.

Figure 2. 8155/8156 On-Board Memory Read/Write Cycle

## PROGRAMMING OF THE COMMAND REGISTER

The command register consists of eight latches. Four bits (0-3) define the mode of the ports, two bits (4-5) enable or disable the interrupt from port C when it acts as control port, and the last two bits (6-7) are for the timer.

The command register contents can be altered at any time by using the I/O address XXXXX000 during a WRITE operation with the Chip Enable active and  $\text{IO}/\overline{\text{M}} = 1$ . The meaning of each bit of the command byte is defined in Figure 3. The contents of the command register may never be read.

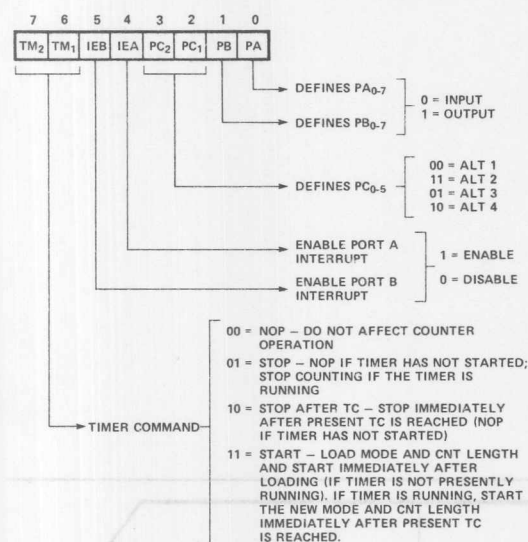


Figure 3. Command Register Bit Assignment

## READING THE STATUS REGISTER

The status register consists of seven latches, one for each bit; six (0-5) for the status of the ports and one (6) for the status of the timer.

The status of the timer and the I/O section can be polled by reading the Status Register (Address XXXXX000). Status word format is shown in Figure 4. Note that you may never write to the status register since the command register shares the same I/O address and the command register is selected when a write to that address is issued.

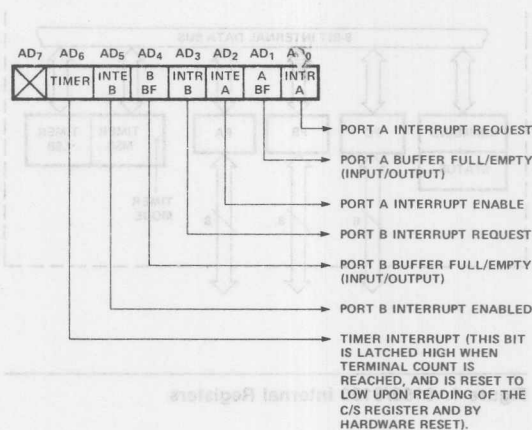


Figure 4. Status Register Bit Assignment

## INPUT/OUTPUT SECTION

The I/O section of the 8155/8156 consists of five registers: (See Figure 5.)

- **Command/Status Register (C/S)** — Both registers are assigned the address XXXXX000. The C/S address serves the dual purpose.

When the C/S registers are selected during WRITE operation, a command is written into the command register. The contents of this register are *not* accessible through the pins.

When the C/S (XXXXX000) is selected during a READ operation, the status information of the I/O ports and the timer becomes available on the AD<sub>0-7</sub> lines.

- **PA Register** — This register can be programmed to be either input or output ports depending on the status of the contents of the C/S Register. Also depending on the command, this port can operate in either the basic mode or the strobed mode (See timing diagram). The I/O pins assigned in relation to this register are PA<sub>0-7</sub>. The address of this register is XXXXX001.
- **PB Register** — This register functions the same as PA Register. The I/O pins assigned are PB<sub>0-7</sub>. The address of this register is XXXXX010.
- **PC Register** — This register has the address XXXXX011 and contains only 6 bits. The 6 bits can be programmed to be either input ports, output ports or as control signals for PA and PB by properly programming the AD<sub>2</sub> and AD<sub>3</sub> bits of the C/S register.

When PC<sub>0-5</sub> is used as a control port, 3 bits are assigned for Port A and 3 for Port B. The first bit is an interrupt that the 8155 sends out. The second is an output signal indicating whether the buffer is full or empty, and the third is an input pin to accept a strobe for the strobed input mode. (See Table 1.)

When the 'C' port is programmed to either ALT3 or ALT4, the control signals for PA and PB are initialized as follows:

CONTROL	INPUT MODE	OUTPUT MODE
BF	Low	Low
INTR	Low	High
STB	Input Control	Input Control

I/O ADDRESS†								SELECTION
A7	A6	A5	A4	A3	A2	A1	A0	
X	X	X	X	X	0	0	0	Interval Command/Status Register
X	X	X	X	X	0	0	1	General Purpose I/O Port A
X	X	X	X	X	0	1	0	General Purpose I/O Port B
X	X	X	X	X	0	1	1	Port C — General Purpose I/O or Control
X	X	X	X	X	1	0	0	Low-Order 8 bits of Timer Count
X	X	X	X	X	1	0	1	High 6 bits of Timer Count and 2 bits of Timer Mode

X: Don't Care.

†: I/O Address must be qualified by CE = 1 (8156) or CE = 0 (8155) and IO/M = 1 in order to select the appropriate register.

Figure 5. I/O port and Timer Addressing Scheme

Figure 6 shows how I/O PORTS A and B are structured within the 8155 and 8156:

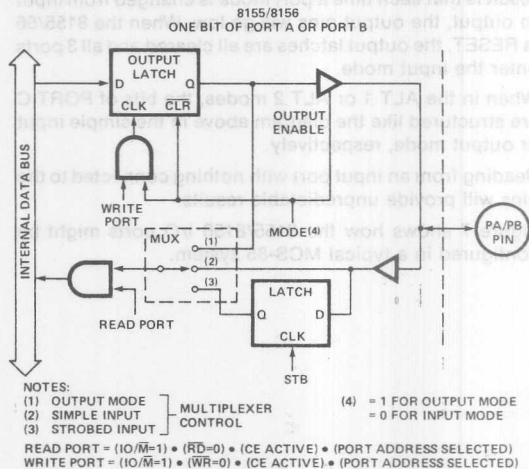


Figure 6. 8155/8156 Port Functions

TABLE 1. TABLE OF PORT CONTROL ASSIGNMENT.

Pin	ALT 1	ALT 2	ALT 3	ALT 4
PC0	Input Port	Output Port	A INTR (Port A Interrupt)	A INTR (Port A Interrupt)
PC1	Input Port	Output Port	A BF (Port A Buffer Full)	A BF (Port A Buffer Full)
PC2	Input Port	Output Port	A STB (Port A Strobe)	A STB (Port A Strobe)
PC3	Input Port	Output Port	Output Port	B INTR (Port B Interrupt)
PC4	Input Port	Output Port	Output Port	B BF (Port B Buffer Full)
PC5	Input Port	Output Port	Output Port	B STB (Port B Strobe)

Note in the diagram that when the I/O ports are programmed to be output ports, the contents of the output ports can still be read by a READ operation when appropriately addressed.

The outputs of the 8155/8156 are "glitch-free" meaning that you can write a "1" to a bit position that was previously "1" and the level at the output pin will not change.

Note also that the output latch is cleared when the port enters the input mode. The output latch cannot be loaded by writing to the port if the port is in the input mode. The result is that each time a port mode is changed from input to output, the output pins will go low. When the 8155/56 is RESET, the output latches are all cleared and all 3 ports enter the input mode.

When in the ALT 1 or ALT 2 modes, the bits of PORT C are structured like the diagram above in the simple input or output mode, respectively.

Reading from an input port with nothing connected to the pins will provide unpredictable results.

Figure 7 shows how the 8155/8156 I/O ports might be configured in a typical MCS-85 system.

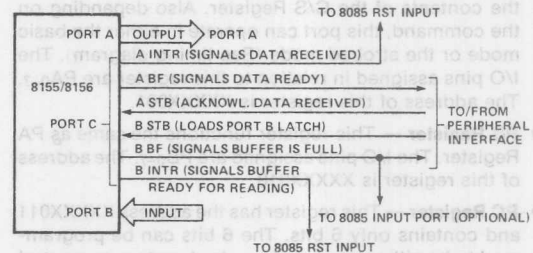
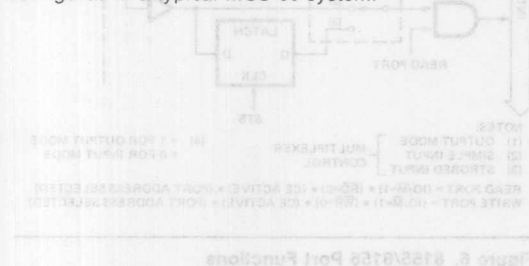


Figure 7. Example: Command Register = 00111001

CONTROL	INPUT MODE	OUTPUT MODE
BF	Low	Low
INTR	Low	High
STB	Input Control	

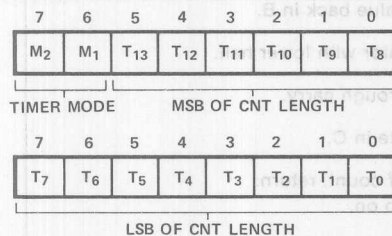


## TIMER SECTION

The timer is a 14-bit down-counter that counts the TIMER IN pulses and provides either a square wave or pulse when terminal count (TC) is reached.

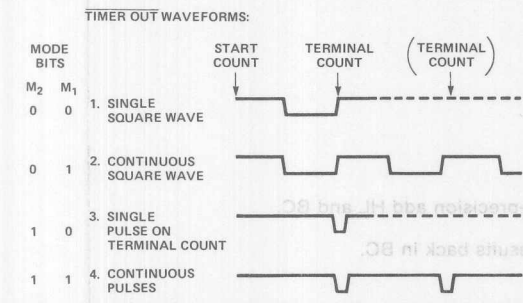
The timer has the I/O address XXXXX100 for the low order byte of the register and the I/O address XXXXX101 for the high order byte of the register. (See Figure 5).

To program the timer, the COUNT LENGTH REG is loaded first, one byte at a time, by selecting the timer addresses. Bits 0-13 of the high order count register will specify the length of the next count and bits 14-15 of the high order register will specify the timer output mode (see Figure 8). The value loaded into the count length register can have any value from 2H through 3FFH in Bits 0-13.



**Figure 8. Timer Format**

There are four modes to choose from: M2 and M1 define the timer mode, as shown in Figure 9.



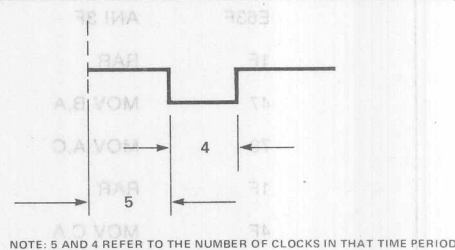
**Figure 9. Timer Modes**

Bits 6-7 (TM<sub>2</sub> and TM<sub>1</sub>) of command register contents are used to start and stop the counter. There are four commands to choose from:

TM <sub>2</sub>	TM <sub>1</sub>	
0	0	NOP — Do not affect counter operation.
0	1	STOP — NOP if timer has not started; stop counting if the timer is running.
1	0	STOP AFTER TC — Stop immediately after present TC is reached (NOP if timer has not started)
1	1	START — Load mode and CNT length and start immediately after loading (if timer is not presently running). If timer is running, start the new mode and CNT length immediately after present TC is reached.

Note that while the counter is counting, you may load a new count and mode into the count length registers. Before the new count and mode will be used by the counter, you must issue a START command to the counter. This applies even though you may only want to change the count and use the previous mode.

In case of an odd-numbered count, the first half-cycle of the squarewave output, which is high, is one count longer than the second (low) half-cycle, as shown in Figure 10.



**Figure 10. Asymmetrical Square-Wave Output Resulting from Count of 9**

The counter in the 8155 is not initialized to any particular mode or count when hardware RESET occurs, but RESET does stop the counting. Therefore, counting cannot begin following RESET until a START command is issued via the C/S register.

Please note that the timer circuit on the 8155/8156 chip is designed to be a square-wave timer, not an event counter. To achieve this, it counts down by two's twice in completing one cycle. Thus, its registers do not contain values directly representing the number of TIMER IN pulses received. You cannot load an initial value of 1 into the count register and cause the timer to operate, as its terminal count value is 10 (binary) or 2 (decimal). (For the detection of single pulses, it is suggested that one of the hardware interrupt pins on the 8085A be used.) After the timer has started counting down, the values residing in the count registers can be used to calculate the actual number of TIMER IN pulses required to complete the timer cycle if desired. To obtain the remaining count, perform the following operations in order:

1. Stop the count
2. Read in the 16-bit value from the count length registers
3. Reset the upper two mode bits
4. Reset the carry and rotate right one position all 16 bits through carry
5. If carry is set, add 1/2 of the full original count (1/2 full count — 1 if full count is odd).

Note: If you started with an odd count and you read the count length register before the third count pulse occurs, you will not be able to discern whether one or two counts has occurred. Regardless of this, the 8155/56 always counts out the right number of pulses in generating the TIMER OUT waveforms.

Following is an actual sequence of program steps that adjusts the 8155/56 count register contents to obtain the count, extracted from Intel® Application Note AP38. "Application Techniques for the Intel 8085A Bus." First store the value of the full original count in register HL of the 8085A. Then stop the count to avoid getting an incorrect count value. Then sample the timer-counter, storing the lower-order byte of the current count register in register C and the higher-order count byte in register B. Then, call the following 8080A/8085A subroutine:

ADJUST, 78      MOV A,B      ;Load accumulator with upper half  
; of count.

E63F      ANI 3F      ;Reset upper 2 bits and clear carry.

1F      RAR      ;Rotate right through carry.

47      MOV B,A      ;Store shifted value back in B.

79      MOV A,C      ;Load accumulator with lower half.

1F      RAR      ;Rotate right through carry.

4F      MOV C,A      ;Store lower byte in C.

D0      RNC      ;If in 2nd half of count, return.  
; If in 1st half, go on.

3F      CMC      ;Clear carry.

7C      MOV A,H      ;Divide full count by 2. (If HL  
; is odd, disregard remainder.)

1F      RAR      ;Divide full count by 2.

67      MOV H,A      ;Divide full count by 2.

7D      MOV A,L      ;Divide full count by 2.

1F      RAR      ;Divide full count by 2.

6F      MOV L,A      ;Divide full count by 2.

09      DAD B      ;Double-precision add HL and BC.

44      MOV B,H      ;Store results back in BC.

4D      MOV C,L      ;Store results back in BC.

C9      RET      ;Return.

After executing the subroutine, BC will contain the remaining count in the current count cycle.

## 8085A MINIMUM SYSTEM CONFIGURATION

Figure 11a shows a minimum system using three chips, containing:

- 256 Bytes RAM
- 2K Bytes ROM
- 38 I/O Pins
- 1 Interval Timer
- 4 Interrupt Levels

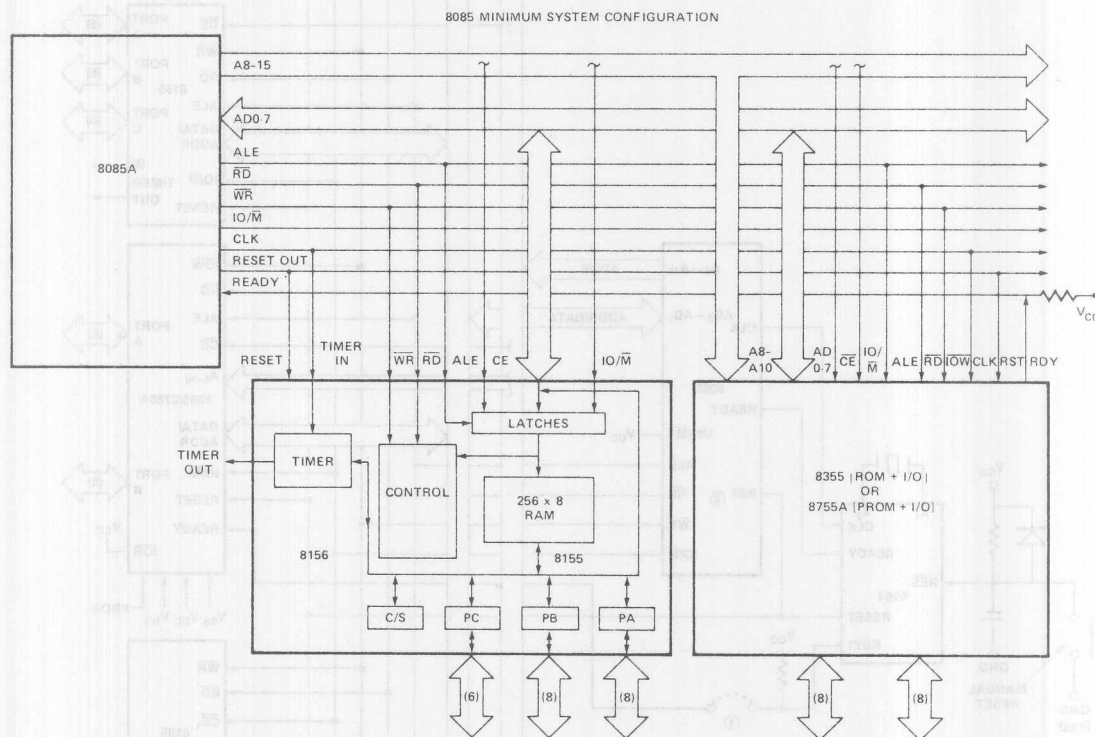


Figure 11a. 8085A Minimum System Configuration. (Memory Mapped I/O)

## 8088 FIVE CHIP SYSTEM

Figure 11b shows a five chip system containing:

- 1.25K Bytes RAM
- 2K Bytes ROM
- 38 I/O Pins
- 1 Interval Timer
- 2 Interrupt Levels

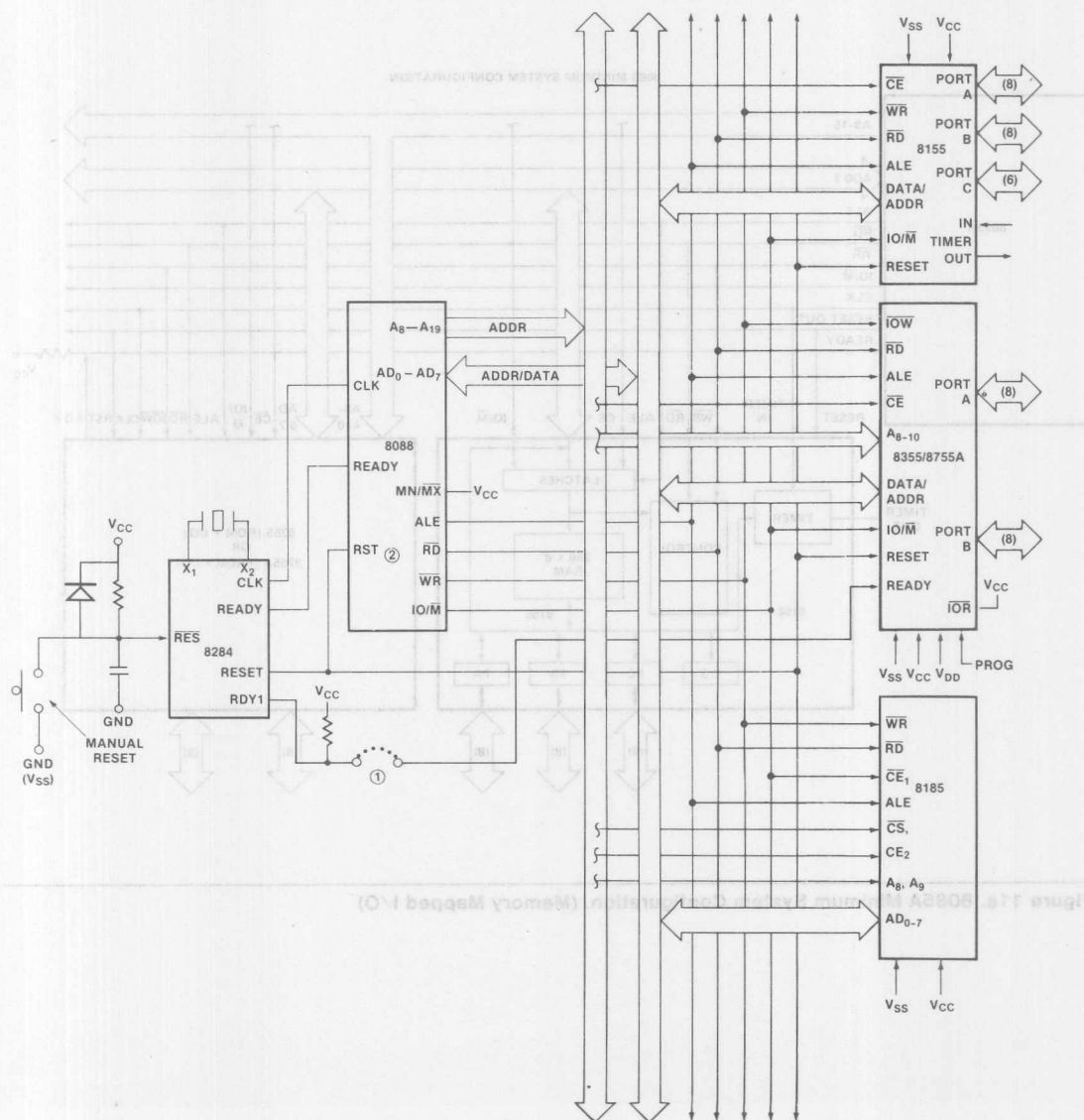


Figure 11b. 8088 Five Chip System Configuration

**ABSOLUTE MAXIMUM RATINGS\***

Temperature Under Bias	0°C to +70°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin With Respect to Ground	-0.5V to +7V
Power Dissipation	1.5W

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

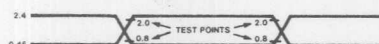
**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ;  $V_{CC} = 5V \pm 5\%$ )

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	
$V_{IH}$	Input High Voltage	2.0	$V_{CC}+0.5$	V	
$V_{OL}$	Output Low Voltage		0.45	V	$I_{OL} = 2\text{mA}$
$V_{OH}$	Output High Voltage	2.4		V	$I_{OH} = -400\mu\text{A}$
$I_{IL}$	Input Leakage		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC} \text{ to } 0V$
$I_{LO}$	Output Leakage Current		$\pm 10$	$\mu\text{A}$	$0.45V \leq V_{OUT} \leq V_{CC}$
$I_{CC}$	$V_{CC}$ Supply Current		180	mA	
$I_{IL} (CE)$	Chip Enable Leakage				
	8155		+100	$\mu\text{A}$	$V_{IN} = V_{CC} \text{ to } 0V$
	8156		-100	$\mu\text{A}$	



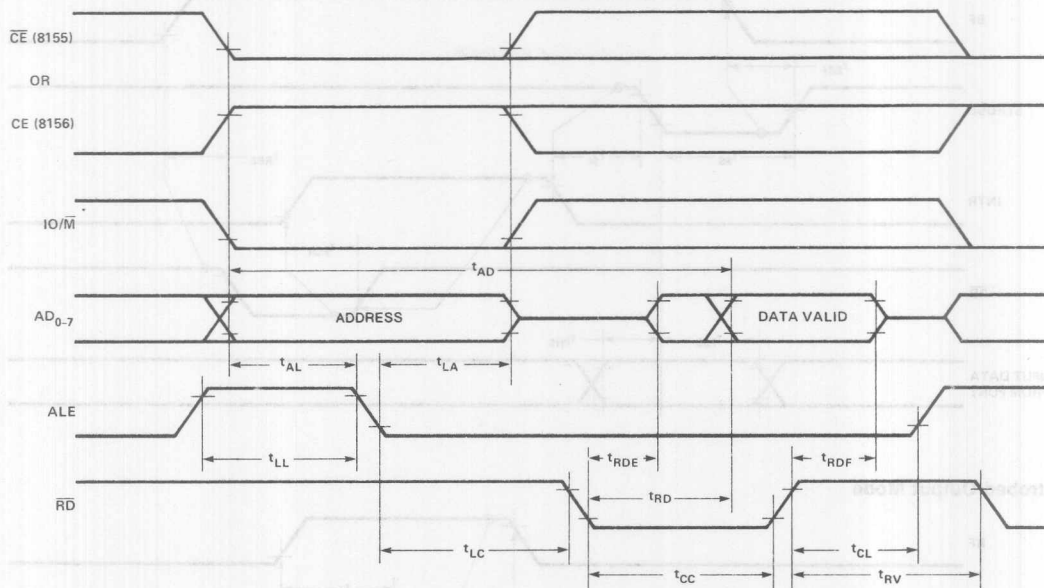
**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5V \pm 5\%$ )

SYMBOL	PARAMETER	8155/8156		8155-2/8156-2 (Preliminary)		UNITS
		MIN.	MAX.	MIN.	MAX.	
$t_{AL}$	Address to Latch Set Up Time	50		30		ns
$t_{LA}$	Address Hold Time after Latch	80		30		ns
$t_{LC}$	Latch to READ/WRITE Control	100		40		ns
$t_{RD}$	Valid Data Out Delay from READ Control		170		140	ns
$t_{AD}$	Address Stable to Data Out Valid		400		330	ns
$t_{LL}$	Latch Enable Width	100		70		ns
$t_{RDF}$	Data Bus Float After READ	0	100	0	80	ns
$t_{CL}$	READ/WRITE Control to Latch Enable	20		10		ns
$t_{CC}$	READ/WRITE Control Width	250		200		ns
$t_{DW}$	Data In to WRITE Set Up Time	150		100		ns
$t_{WD}$	Data In Hold Time After WRITE	0		0		ns
$t_{RV}$	Recovery Time Between Controls	300		200		ns
$t_{WP}$	WRITE to Port Output		400		300	ns
$t_{PR}$	Port Input Setup Time	70		50		ns
$t_{RP}$	Port Input Hold Time	50		10		ns
$t_{SBF}$	Strobe to Buffer Full		400		300	ns
$t_{SS}$	Strobe Width	200		150		ns
$t_{RBE}$	READ to Buffer Empty		400		300	ns
$t_{SI}$	Strobe to INTR On		400		300	ns
$t_{RDI}$	READ to INTR Off		400		300	ns
$t_{PSS}$	Port Setup Time to Strobe Strobe	50		0		ns
$t_{PHS}$	Port Hold Time After Strobe	120		100		ns
$t_{SBE}$	Strobe to Buffer Empty		400		300	ns
$t_{WBF}$	WRITE to Buffer Full		400		300	ns
$t_{WI}$	WRITE to INTR Off		400		300	ns
$t_{TL}$	TIMER-IN to $\overline{\text{TIMER-OUT}}$ Low		400		300	ns
$t_{TH}$	TIMER-IN to $\overline{\text{TIMER-OUT}}$ High		400		300	ns
$t_{RDE}$	Data Bus Enable from READ Control	10		10		ns
$t_1$	TIMER-IN Low Time	80		40		ns
$t_2$	TIMER-IN High Time	120		70		ns

**Input Waveform for A.C. Tests:**

## WAVEFORMS

## a. Read Cycle



## b. Write Cycle

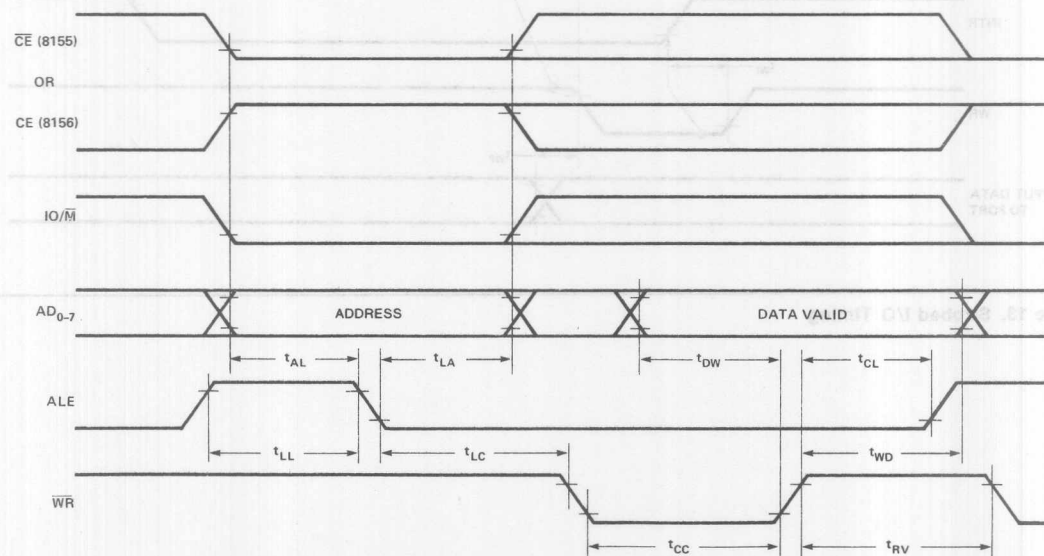
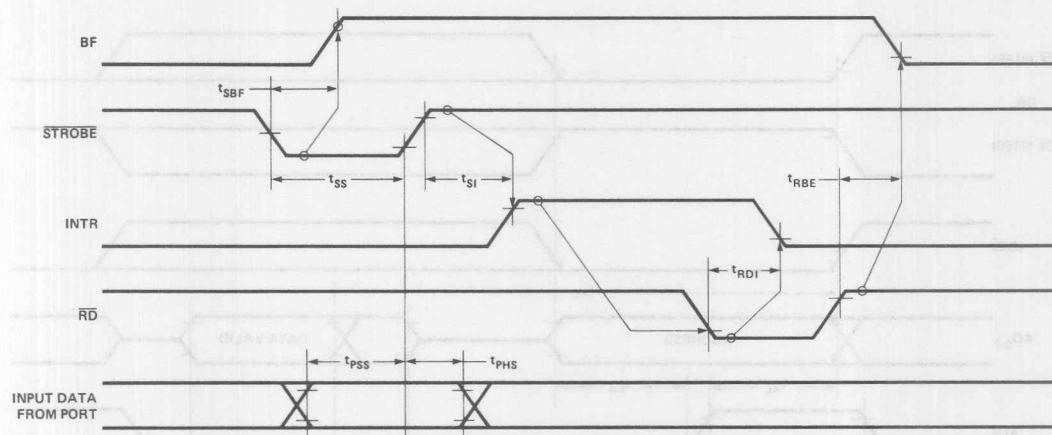


Figure 12. 8155/8156 Read/Write Timing Diagrams

## a. Strobed Input Mode



## b. Strobed Output Mode

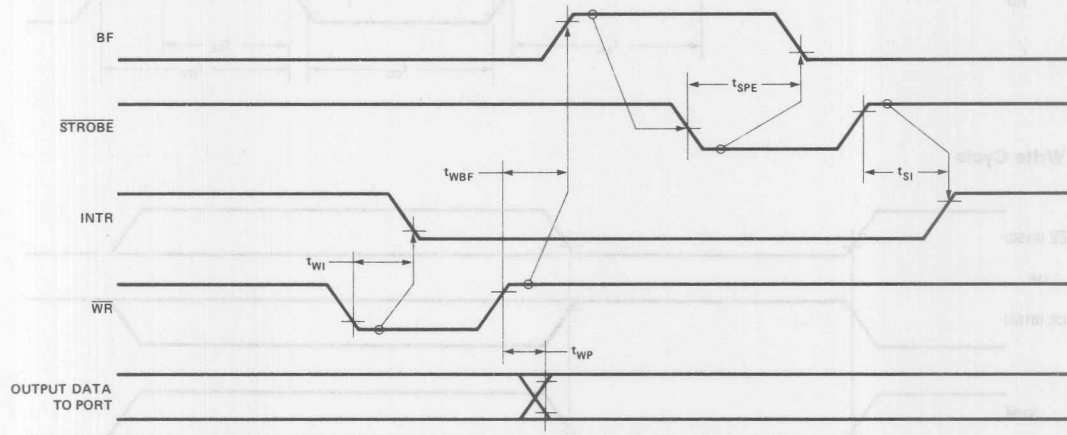
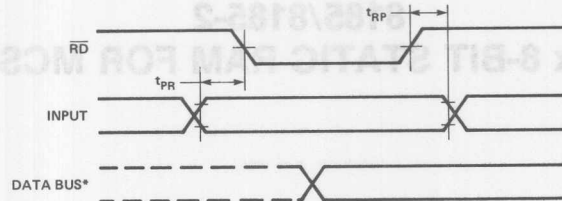
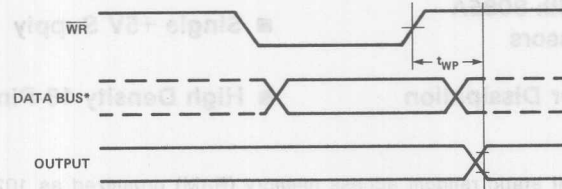


Figure 13. Strobed I/O Timing

## a. Basic Input Mode



## b. Basic Output Mode



\*DATA BUS TIMING IS SHOWN IN FIGURE 7.

Figure 14. Basic I/O Timing Waveform

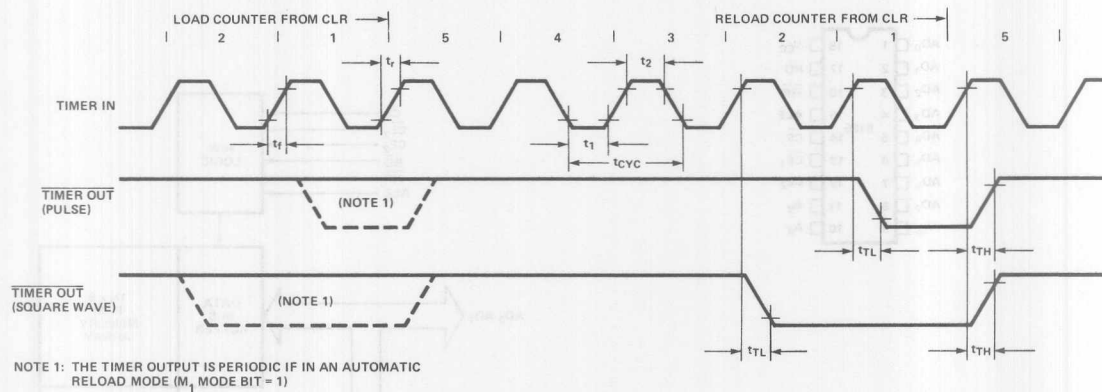


Figure 15. Timer Output Waveform Countdown from 5 to 1

# 8185/8185-2

## 1024 x 8-BIT STATIC RAM FOR MCS-85™

Intel is a final specification. Some details are subject to change.

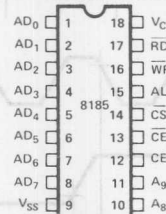
- Multiplexed Address and Data Bus
- Low Standby Power Dissipation
- Directly Compatible with 8085A and 8088 Microprocessors
- Single +5V Supply
- Low Operating Power Dissipation
- High Density 18-Pin Package

The Intel® 8185 is an 8192-bit static random access memory (RAM) organized as 1024 words by 8-bits using N-channel Silicon-Gate MOS technology. The multiplexed address and data bus allows the 8185 to interface directly to the 8085A and 8088 microprocessors to provide a maximum level of system integration.

The low standby power dissipation minimizes system power requirements when the 8185 is disabled.

The 8185-2 is a high-speed selected version of the 8185 that is compatible with the 5 MHz 8085A-2 and the full speed 5 MHz 8088.

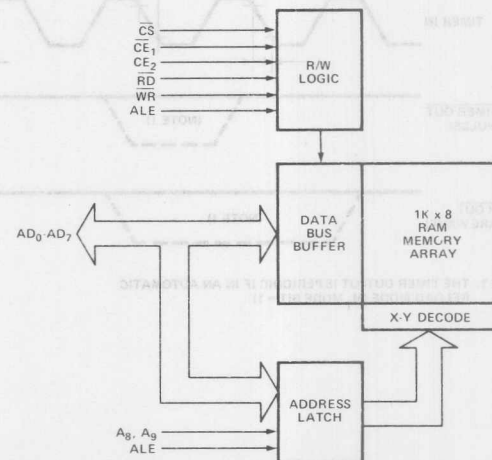
### PIN CONFIGURATION



### PIN NAMES

AD <sub>0</sub> -AD <sub>7</sub>	ADDRESS/DATA LINES
A <sub>8</sub> , A <sub>9</sub>	ADDRESS LINES
CS	CHIP SELECT
CE <sub>1</sub>	CHIP ENABLE (IO/ $\overline{M}$ )
CE <sub>2</sub>	CHIP ENABLE
ALE	ADDRESS LATCH ENABLE
RD	READ ENABLE
WR	WRITE ENABLE

### BLOCK DIAGRAM









**ABSOLUTE MAXIMUM RATINGS\***

Temperature Under Bias ..... 0°C to +70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage on Any Pin  
   with Respect to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 1.5W

**\*COMMENT**

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5V \pm 5\%$ )

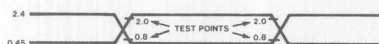
Symbol	Parameter	Min.	Max.	Units	Test Conditions
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	
$V_{IH}$	Input High Voltage	2.0	$V_{CC}+0.5$	V	
$V_{OL}$	Output Low Voltage		0.45	V	$I_{OL} = 2\text{mA}$
$V_{OH}$	Output High Voltage	2.4		V	$I_{OH} = 400\mu\text{A}$
$I_{IL}$	Input Leakage		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ to 0V
$I_{LO}$	Output Leakage Current		$\pm 10$	$\mu\text{A}$	$0.45V \leq V_{OUT} \leq V_{CC}$
$I_{CC}$	$V_{CC}$ Supply Current Powered Up		100	mA	
	Powered Down		25	mA	

**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5V \pm 5\%$ )

Symbol	Parameter <sup>[1]</sup>	8185 Preliminary		8185-2 Preliminary		Units
		Min.	Max.	Min.	Max.	
$t_{AL}$	Address to Latch Set Up Time	50		30		ns
$t_{LA}$	Address Hold Time After Latch	80		30		ns
$t_{LC}$	Latch to READ/WRITE Control	100		40		ns
$t_{RD}$	Valid Data Out Delay from READ Control		170		140	ns
$t_{LD}$	ALE to Data Out Valid		300		200	ns
$t_{LL}$	Latch Enable Width	100		70		ns
$t_{RDF}$	Data Bus Float After READ	0	100	0	80	ns
$t_{CL}$	READ/WRITE Control to Latch Enable	20		10		ns
$t_{CC}$	READ/WRITE Control Width	250		200		ns
$t_{DW}$	Data In to WRITE Set Up Time	150		150		ns
$t_{WD}$	Data In Hold Time After WRITE	20		20		ns
$t_{SC}$	Chip Select Set Up to Control Line	10		10		ns
$t_{CS}$	Chip Select Hold Time After Control	10		10		ns
$t_{ALCE}$	Chip Enable Set Up to ALE Falling	30		10		ns
$t_{LACE}$	Chip Enable Hold Time After ALE	50		30		ns

**Notes:**

1. All AC parameters are referenced at
  - a) 2.4V and .45V for inputs
  - b) 2.0V and .8V for outputs.

**Input Waveform for A.C. Tests:**

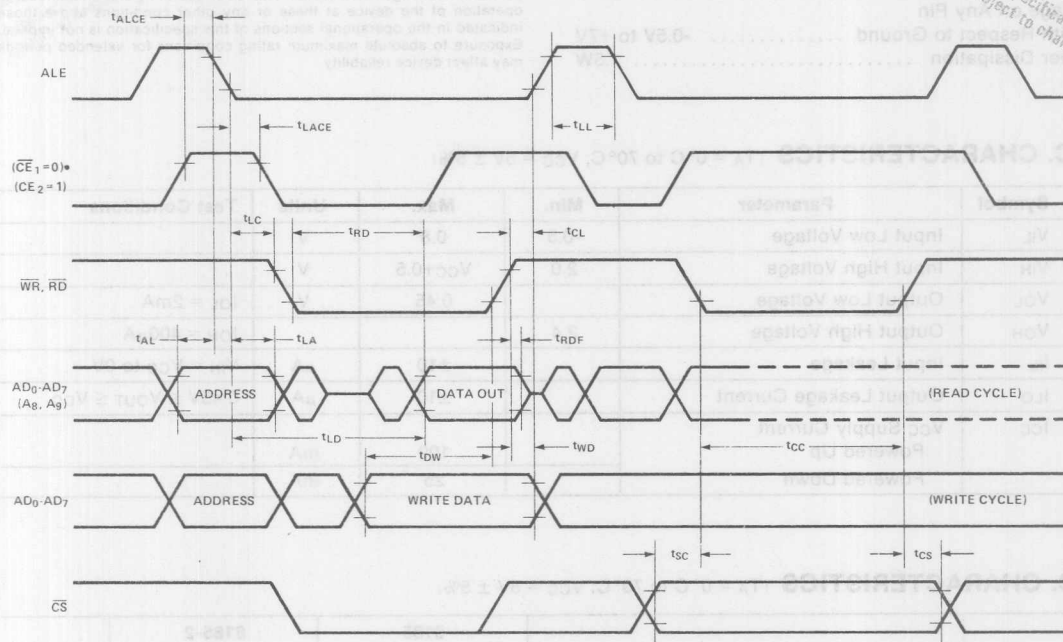


Figure 3. 8185 Timing.

Symbol	Parameter	(SELECTED)		(DESELECTED)	
		Min.	Max.	Min.	Max.
$t_{AL}$	Address to Latch Set Up Time	50		50	
$t_{LA}$	Address Hold Time After Latch	80		80	
$t_{LC}$	Latch to READ/WRITE Control	100		100	
$t_{RD}$	Valid Data Out Delay from READ Control	110		110	
$t_{LD}$	ALE to Data Out Valid	300		300	
$t_{LL}$	Latch Enable Width	70		70	
$t_{RD}$	Data Bus First After READ	0	100	0	100
$t_{CL}$	READ/WRITE Control to Latch Enable	20		20	
$t_{OW}$	READ/WRITE Control Width	200		200	
$t_{WD}$	Data In to WRITE Set Up Time	150		150	
$t_{OW}$	Data In Hold Time After WRITE	20		20	
$t_{sc}$	Chip Select Set Up to Control Line	10		10	
$t_{cs}$	Chip Select Hold Time After Control	10		10	
$t_{ALCE}$	Chip Enable Set Up to ALE Falling	10		10	
$t_{LACE}$	Chip Enable Hold Time After ALE	30		30	

Notes:  
 1. All AC parameters are referenced at:  
 a) 2.4V and 4.5V for inputs  
 b) 2.0V and 3V for outputs  
 Input Waveform for A.C. Tests:



# 8355/8355-2

## 16,384-BIT ROM WITH I/O

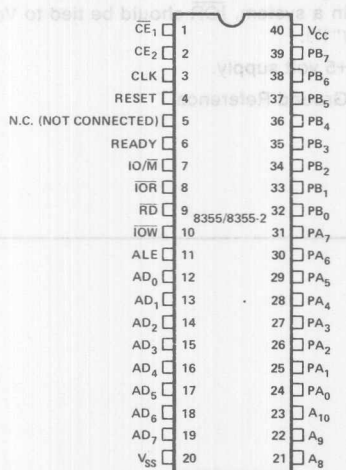
- 2048 Words x 8 Bits
- Single +5V Power Supply
- Directly compatible with 8085A and 8088 Microprocessors
- 2 General Purpose 8-Bit I/O Ports
- Each I/O Port Line Individually Programmable as Input or Output
- Multiplexed Address and Data Bus
- Internal Address Latch
- 40-Pin DIP

The Intel® 8355 is a ROM and I/O chip to be used in the 8085A and 8088 microprocessor systems. The ROM portion is organized as 2048 words by 8 bits. It has a maximum access time of 400 ns to permit use with no wait states in the 8085A CPU.

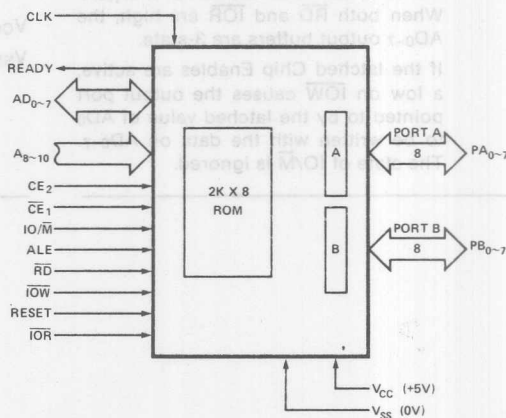
The I/O portion consists of 2 general purpose I/O ports. Each I/O port has 8 port lines and each I/O port line is individually programmable as input or output.

The 8355-2 has a 300ns access time for compatibility with the 8085A-2 and full speed 5 MHz 8088 microprocessors.

### PIN CONFIGURATION

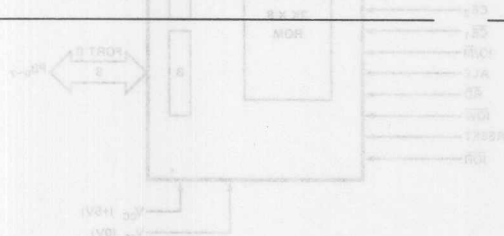


### BLOCK DIAGRAM





Symbol	Function	Symbol	Function
ALE (Input)	When ALE (Address Latch Enable is high, AD <sub>0-7</sub> , IO/ $\overline{M}$ , A <sub>8-10</sub> , CE <sub>1</sub> and $\overline{CE}$ enter address latched. The signals (AD, IO/ $\overline{M}$ , A <sub>8-10</sub> , CE, $\overline{CE}$ ) are latched in at the trailing edge of ALE.	CLK (Input)	The CLK is used to force the READY into its high impedance state after it has been forced low by $\overline{CE}$ low, CE high and ALE high.
AD <sub>0-7</sub> (Input)	Bidirectional Address/Data bus. The lower 8-bits of the ROM or I/O address are applied to the bus lines when ALE is high.  During an I/O cycle, Port A or B are selected based on the latched value of AD <sub>0</sub> . If $\overline{RD}$ or $\overline{IOR}$ is low when the latched chip enables are active, the output buffers present data on the bus.	READY (Output)	Ready is a 3-state output controlled by $\overline{CE}$ <sub>1</sub> , CE <sub>2</sub> , ALE and CLK. READY is forced low when the Chip Enables are active during the time ALE is high, and remains low until the rising edge of the next CLK (see Figure 6).
A <sub>8-10</sub> (Input)	These are the high order bits of the ROM address. They do not affect I/O operations.	PA <sub>0-7</sub> (Input/ Output)	These are general purpose I/O pins. Their input/output direction is determined by the contents of Data Direction Register (DDR). Port A is selected for write operations when the Chip Enables are active and $\overline{IOW}$ is low and a 0 was previously latched from AD <sub>0</sub> .
$\overline{CE}$ <sub>1</sub> CE <sub>2</sub> (Input)	Chip Enable Inputs: $\overline{CE}$ <sub>1</sub> is active low and CE <sub>2</sub> is active high. The 8355 can be accessed only when BOTH Chip Enables are active at the time the ALE signal latches them up. If either Chip Enable input is not active, the AD <sub>0-7</sub> and READY outputs will be in a high impedance state.		Read operation is selected by either $\overline{IOR}$ low and active Chip Enables and AD <sub>0</sub> low, or IO/ $\overline{M}$ high, $\overline{RD}$ low, active chip enables, and AD <sub>0</sub> low.
IO/ $\overline{M}$ (Input)	If the latched IO/ $\overline{M}$ is high when $\overline{RD}$ is low, the output data comes from an I/O port. If it is low the output data comes from the ROM.	PB <sub>0-7</sub> (Input/ Output)	This general purpose I/O port is identical to Port A except that it is selected by a 1 latched from AD <sub>0</sub> .
$\overline{RD}$ (Input)	If the latched Chip Enables are active when $\overline{RD}$ goes low, the AD <sub>0-7</sub> output buffers are enabled and output either the selected ROM location or I/O port. When both $\overline{RD}$ and $\overline{IOR}$ are high, the AD <sub>0-7</sub> output buffers are 3-state.	RESET (Input)	An input high on RESET causes all pins in Port A and B to assume input mode.
$\overline{IOW}$ (Input)	If the latched Chip Enables are active, a low on $\overline{IOW}$ causes the output port pointed to by the latched value of AD <sub>0</sub> to be written with the data on AD <sub>0-7</sub> . The state of IO/ $\overline{M}$ is ignored.	$\overline{IOR}$ (Input)	When the Chip Enables are active, a low on $\overline{IOR}$ will output the selected I/O port onto the AD bus. $\overline{IOR}$ low performs the same function as the combination IO/ $\overline{M}$ high and $\overline{RD}$ low. When $\overline{IOR}$ is not used in a system, $\overline{IOR}$ should be tied to V <sub>CC</sub> ("1").
		V <sub>CC</sub>	+5 volt supply.
		V <sub>SS</sub>	Ground Reference.



## FUNCTIONAL DESCRIPTION

## ROM Section

The 8355 contains an 8-bit address latch which allows it to interface directly to MCS-48 and MCS-85 Microcomputers without additional hardware.

The ROM section of the chip is addressed by an 11-bit address and the Chip Enables. The address and levels on the Chip Enable pins are latched into the address latches on the falling edge of ALE. If the latched Chip Enables are active and IO/M is low when RD goes low, the contents of the ROM location addressed by the latched address are put out through AD0-7 output buffers.

## I/O Section

The I/O section of the chip is addressed by the latched value of AD<sub>0-1</sub>. Two 8-bit Data Direction Registers (DDR) in 8355 determine the input/output status of each pin in the corresponding ports. A "0" in a particular bit position of a DDR signifies that the corresponding I/O port bit is in the input mode. A "1" in a particular bit position signifies that the corresponding I/O port bit is in the output mode. In this manner the I/O ports of the 8355 are bit-by-bit programmable as inputs or outputs. The table summarizes port and DDR designation. DDR's cannot be read.

AD <sub>1</sub>	AD <sub>0</sub>	Selection
0	0	Port A
0	1	Port B
1	0	Port A Data Direction Register (DDR A)
1	1	Port B Data Direction Register (DDR B)

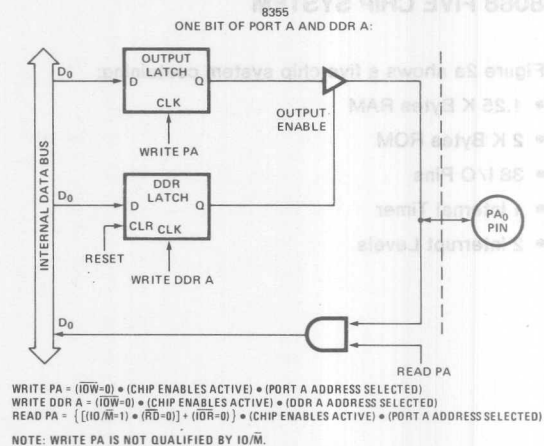
When  $\overline{\text{IOW}}$  goes low and the Chip Enables are active, the data on the  $\text{AD}_{0-7}$  is written into I/O port selected by the latched value of  $\text{AD}_{0-1}$ . During this operation all I/O bits of the selected port are affected, regardless of their I/O mode and the state of  $\text{IO}/\overline{\text{M}}$ . The actual output level does not change until  $\overline{\text{IOW}}$  returns high (glitch free output).

A port can be read out when the latched Chip Enables are active and either  $\overline{\text{RD}}$  goes low with  $\text{IO}/\overline{\text{M}}$  high, or  $\overline{\text{IOR}}$  goes low. Both input and output mode bits of a selected port will appear on lines AD<sub>0-7</sub>.

To clarify the function of the I/O ports and Data Direction Registers, the following diagram shows the configuration of one bit of PORT A and DDR A. The same logic applies to PORT B and DDR B.

Note that hardware RESET or writing a zero to the DDR latch will cause the output latch's output buffer to be disabled, preventing the data in the output latch from being passed through to the pin. This is equivalent to putting the port in the input mode. Note also that the data can be written to the Output Latch even though the Output Buffer has been disabled. This enables a port to be initialized with a value prior to enabling the output.

The diagram also shows that the contents of PORT A and PORT B can be read even when the ports are configured as outputs.



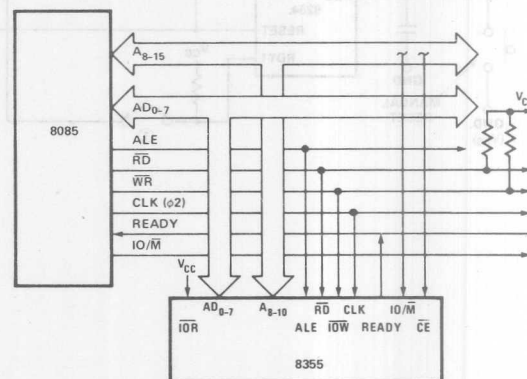
## System Interface with 8085A and 8088

A system using the 8355 can use either one of the two I/O Interface techniques:

- Standard I/O
- Memory Mapped I/O

If a standard I/O technique is used, the system can use the feature of both CE and  $\overline{\text{CE}}$ . By using a combination of unused address lines A<sub>11-15</sub> and the Chip Enable inputs, the system can use up to 5 each 8355's without requiring a CE decoder. See Figure 2a and 2b.

If a memory mapped I/O approach is used the 8355 will be selected by the combination of both the Chip Enables and IO/ $\overline{M}$  using the AD<sub>8-15</sub> address lines. See Figure 1.



**Figure 1. 8355 in 8085A System (Memory-Mapped I/O)**

## 8088 FIVE CHIP SYSTEM

Figure 2a shows a five chip system containing:

- 1.25 K Bytes RAM
- 2 K Bytes ROM
- 38 I/O Pins
- 1 Internal Timer
- 2 Interrupt Levels

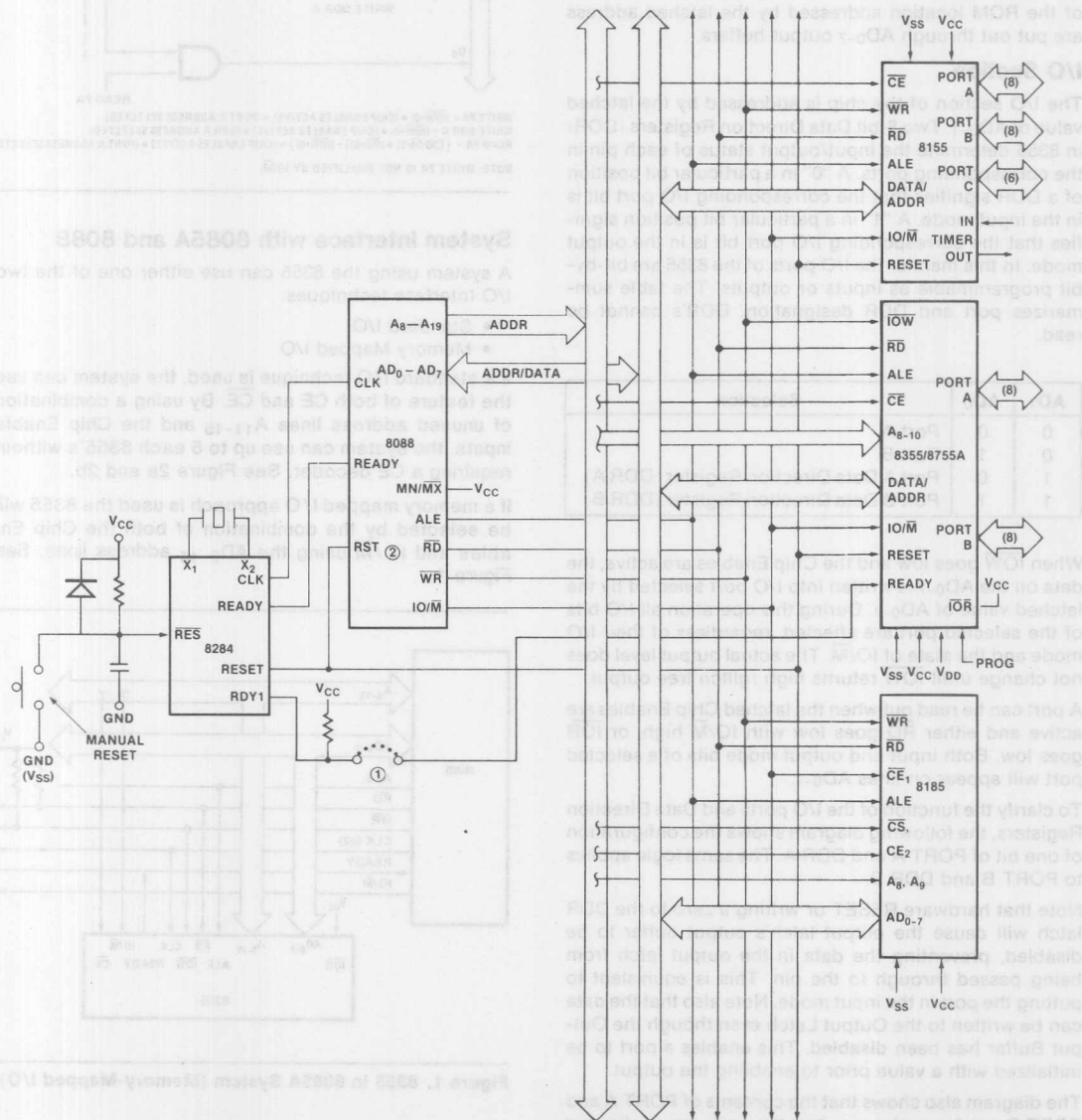
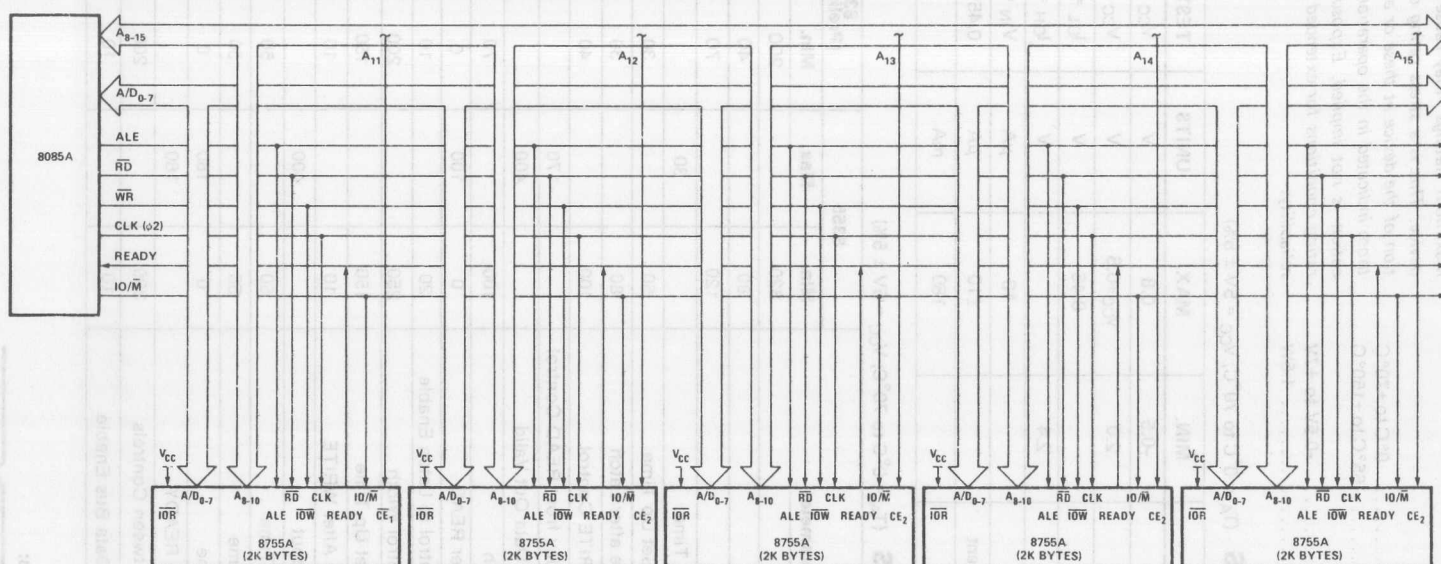


Figure 2a. 8088 Five Chip System Configuration



Note: Use  $\overline{CE}_1$  for the first 8755A in the system, and  $\overline{CE}_2$  for the other 8755A's. Permits up to 5-8755A's in a system without CE decoder.

Figure 2b. 8355 in 8085A System (Standard I/O)

**ABSOLUTE MAXIMUM RATINGS\***

Temperature Under Bias .....	0°C to +70°C
Storage Temperature .....	-65°C to +150°C
Voltage on Any Pin	
With Respect to Ground .....	-0.5V to +7V
Power Dissipation .....	1.5W

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

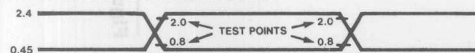
**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5V \pm 5\%$ )

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	$V_{CC} = 5.0V$
$V_{IH}$	Input High Voltage	2.0	$V_{CC}+0.5$	V	$V_{CC} = 5.0V$
$V_{OL}$	Output Low Voltage		0.45	V	$I_{OL} = 2mA$
$V_{OH}$	Output High Voltage	2.4		V	$I_{OH} = -400\mu A$
$I_{IL}$	Input Leakage		10	$\mu A$	$V_{IN} = V_{CC}$ to 0V
$I_{LO}$	Output Leakage Current		$\pm 10$	$\mu A$	$0.45V \leq V_{OUT} \leq V_{CC}$
$I_{CC}$	$V_{CC}$ Supply Current		180	mA	

**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5V \pm 5\%$ )

Symbol	Parameter	8355		8355-2 (Preliminary)		Units
		Min.	Max.	Min.	Max.	
$t_{CYC}$	Clock Cycle Time	320		200		ns
$T_1$	CLK Pulse Width	80		40		ns
$T_2$	CLK Pulse Width	120		70		ns
$t_r, t_f$	CLK Rise and Fall Time		30		30	ns
$t_{AL}$	Address to Latch Set Up Time	50		30		ns
$t_{LA}$	Address Hold Time after Latch	80		30		ns
$t_{LC}$	Latch to READ/WRITE Control	100		40		ns
$t_{RD}$	Valid Data Out Delay from READ Control		170		140	ns
$t_{AD}$	Address Stable to Data Out Valid		400		330	ns
$t_{LL}$	Latch Enable Width	100		70		ns
$t_{RDF}$	Data Bus Float after READ	0	100	0	85	ns
$t_{CL}$	READ/WRITE Control to Latch Enable	20		10		ns
$t_{CC}$	READ/WRITE Control Width	250		200		ns
$t_{DW}$	Data In to Write Set Up Time	150		150		ns
$t_{WD}$	Data In Hold Time After WRITE	10		10		ns
$t_{WP}$	WRITE to Port Output		400		400	ns
$t_{PR}$	Port Input Set Up Time	50		50		ns
$t_{RP}$	Port Input Hold Time	50		50		ns
$t_{RYH}$	READY HOLD Time	0	160	0	160	ns
$t_{ARY}$	ADDRESS (CE) to READY		160		160	ns
$t_{RV}$	Recovery Time Between Controls	300		200		ns
$t_{RDE}$	READ Control to Data Bus Enable	10		10		ns

Note:  $C_{LOAD} = 150pF$

**Input Waveform for A.C. Tests:**



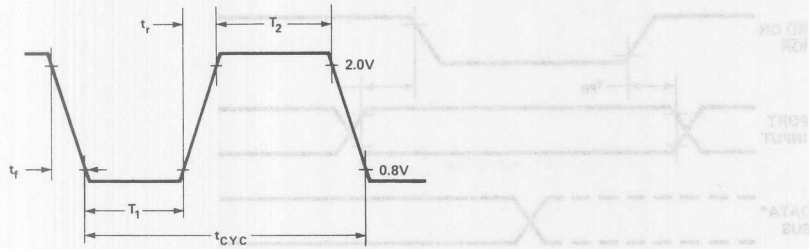


Figure 3. Clock Specification for 8355

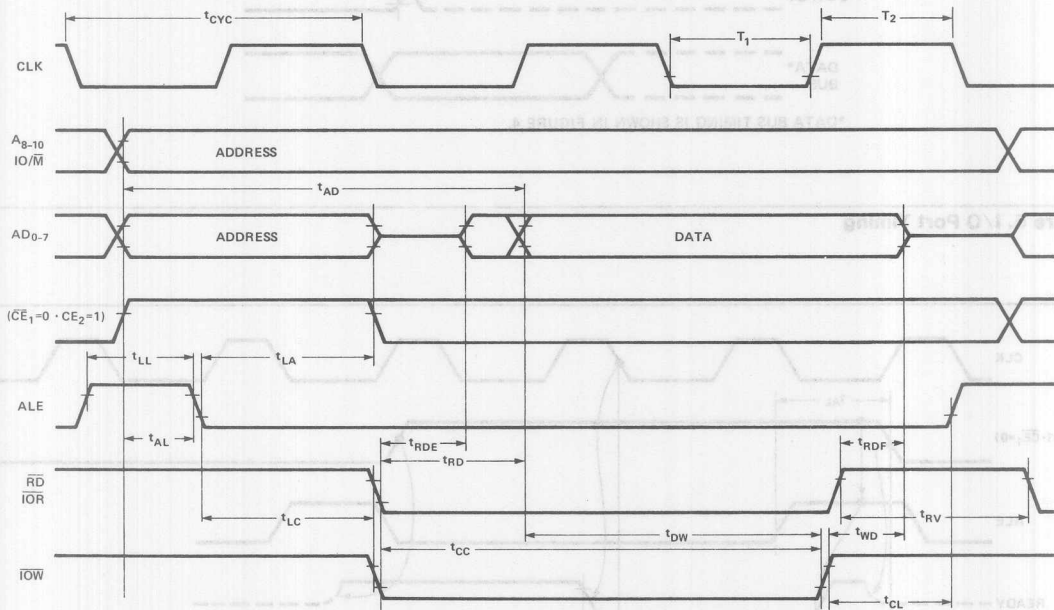
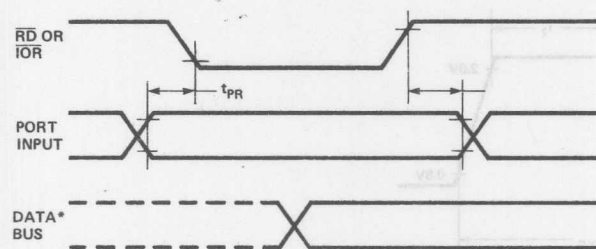
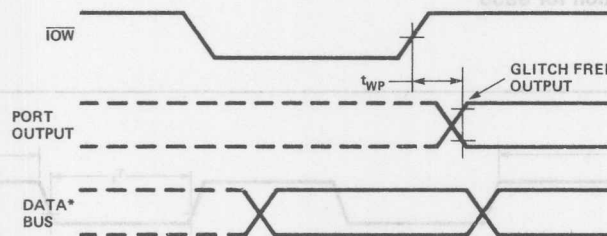


Figure 4. ROM Read and I/O Read and Write

## a. Input Mode



## b. Output Mode



\*DATA BUS TIMING IS SHOWN IN FIGURE 4.

Figure 5. I/O Port Timing

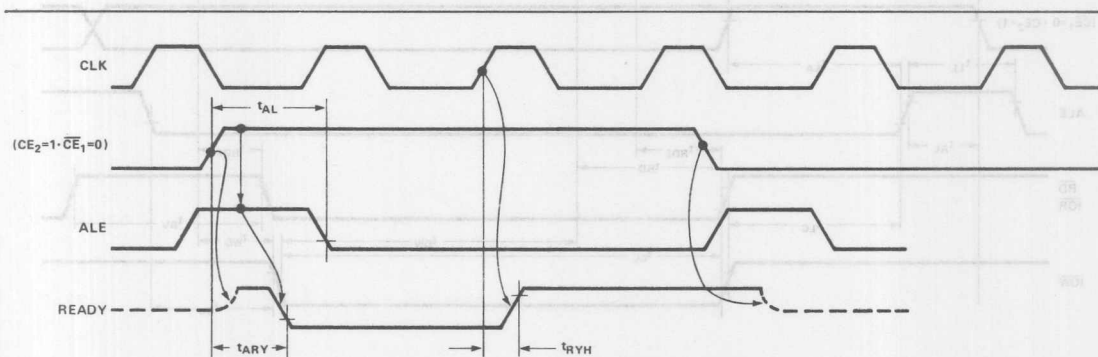


Figure 6. Wait State Timing (Ready = 0)

# 8755A/8755A-2 16,384-BIT EPROM WITH I/O

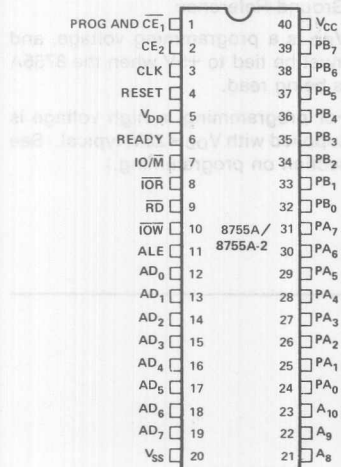
- 2048 Words x 8 Bits
- Single +5V Power Supply ( $V_{CC}$ )
- Directly Compatible with 8085A and 8088 Microprocessors
- U.V. Erasable and Electrically Reprogrammable
- Internal Address Latch
- 2 General Purpose 8-Bit I/O Ports
- Each I/O Port Line Individually Programmable as Input or Output
- Multiplexed Address and Data Bus
- 40-Pin DIP

The Intel® 8755A is an erasable and electrically reprogrammable ROM (EPROM) and I/O chip to be used in the 8085A and 8088 microprocessor systems. The EPROM portion is organized as 2048 words by 8 bits. It has a maximum access time of 450 ns to permit use with no wait states in an 8085A CPU.

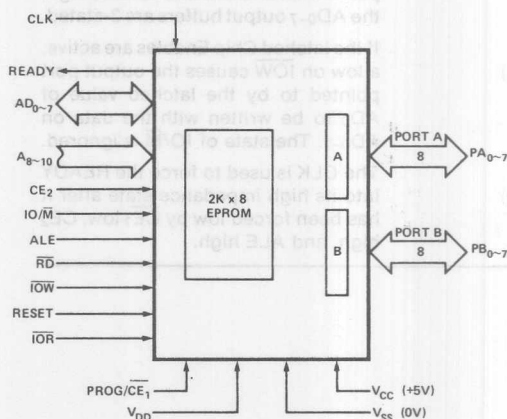
The I/O portion consists of 2 general purpose I/O ports. Each I/O port has 8 port lines, and each I/O port line is individually programmable as input or output.

The 8755A-2 is a high speed selected version of the 8755A compatible with the 5 MHz 8085A-2 and the full speed 5 MHz 8088.

## PIN CONFIGURATION



## BLOCK DIAGRAM



## 8755A FUNCTIONAL PIN DEFINITION

Symbol	Function	Symbol	Function
ALE (input)	When Address Latch Enable goes high, AD <sub>0-7</sub> , IO/M, A <sub>8-10</sub> , CE <sub>2</sub> , and CE <sub>1</sub> enter the address latches. The signals (AD, IO/M, A <sub>8-10</sub> , CE) are latched in at the trailing edge of ALE.	READY (output)	READY is a 3-state output controlled by CE <sub>2</sub> , CE <sub>1</sub> , ALE and CLK. READY is forced low when the Chip Enables are active during the time ALE is high, and remains low until the rising edge of the next CLK. (See Figure 6.)
AD <sub>0-7</sub> (input/output)	Bidirectional Address/Data bus. The lower 8-bits of the PROM or I/O address are applied to the bus lines when ALE is high.  During an I/O cycle, Port A or B are selected based on the latched value of AD <sub>0</sub> . If $\overline{RD}$ or $\overline{IOR}$ is low when the latched Chip Enables are active, the output buffers present data on the bus.	PA <sub>0-7</sub> (input/output)	These are general purpose I/O pins. Their input/output direction is determined by the contents of Data Direction Register (DDR). Port A is selected for write operations when the Chip Enables are active and $\overline{IOW}$ is low and a 0 was previously latched from AD <sub>0</sub> , AD <sub>1</sub> .  Read operation is selected by either $\overline{IOR}$ low and active Chip Enables and AD <sub>0</sub> and AD <sub>1</sub> low, or IO/M high, $\overline{RD}$ low, active Chip Enables, and AD <sub>0</sub> and AD <sub>1</sub> low.
A <sub>8-10</sub> (input)	These are the high order bits of the PROM address. They do not affect I/O operations.	PB <sub>0-7</sub> (input/output)	This general purpose I/O port is identical to Port A except that it is selected by a 1 latched from AD <sub>0</sub> and a 0 from AD <sub>1</sub> .
PROG/ $\overline{CE_1}$ CE <sub>2</sub> (input)	Chip Enable Inputs: $\overline{CE_1}$ is active low and CE <sub>2</sub> is active high. The 8755A can be accessed only when BOTH Chip Enables are active at the time the ALE signal latches them up. If either Chip Enable input is not active, the AD <sub>0-7</sub> and READY outputs will be in a high impedance state. $\overline{CE_1}$ is also used as a programming pin. (See section on programming.)	RESET (input)	In normal operation, an input high on RESET causes all pins in Ports A and B to assume input mode (clear DDR register).
IO/ $\overline{M}$ (input)	If the latched IO/ $\overline{M}$ is high when $\overline{RD}$ is low, the output data comes from an I/O port. If it is low the output data comes from the PROM.	$\overline{IOR}$ (input)	When the Chip Enables are active, a low on $\overline{IOR}$ will output the selected I/O port onto the AD bus. $\overline{IOR}$ low performs the same function as the combination of IO/M high and $\overline{RD}$ low. When $\overline{IOR}$ is not used in a system, $\overline{IOR}$ should be tied to V <sub>CC</sub> ("1").
$\overline{RD}$ (input)	If the latched Chip Enables are active when $\overline{RD}$ goes low, the AD <sub>0-7</sub> output buffers are enabled and output either the selected PROM location or I/O port. When both $\overline{RD}$ and $\overline{IOR}$ are high, the AD <sub>0-7</sub> output buffers are 3-stated.	V <sub>CC</sub>	+5 volt supply.
$\overline{IOW}$ (input)	If the latched Chip Enables are active, a low on $\overline{IOW}$ causes the output port pointed to by the latched value of AD <sub>0</sub> to be written with the data on AD <sub>0-7</sub> . The state of IO/ $\overline{M}$ is ignored.	V <sub>SS</sub>	Ground Reference.
CLK (input)	The CLK is used to force the READY into its high impedance state after it has been forced low by $\overline{CE_1}$ low, CE <sub>2</sub> high, and ALE high.	V <sub>DD</sub>	V <sub>DD</sub> is a programming voltage, and must be tied to +5V when the 8755A is being read.  For programming, a high voltage is supplied with V <sub>DD</sub> = 25V, typical. (See section on programming.)

## FUNCTIONAL DESCRIPTION

## PROM Section

The 8755A contains an 8-bit address latch which allows it to interface directly to MCS-48 and MCS-85 Micro-computers without additional hardware.

The PROM section of the chip is addressed by the 11-bit address and CE. The address,  $\overline{CE}_1$  and  $\overline{CE}_2$  are latched into the address latches on the falling edge of ALE. If the latched Chip Enables are active and IO/M is low when  $\overline{RD}$  goes low, the contents of the PROM location addressed by the latched address are put out on the AD<sub>0-7</sub> lines.

## I/O Section

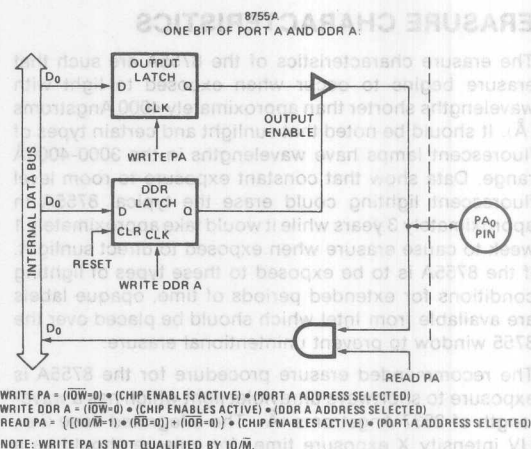
The I/O section of the chip is addressed by the latched value of AD<sub>0-1</sub>. Two 8-bit Data Direction Registers (DDR) in 8755A determine the input/output status of each pin in the corresponding ports. A "0" in a particular bit position of a DDR signifies that the corresponding I/O port bit is in the input mode. A "1" in a particular bit position signifies that the corresponding I/O port bit is in the output mode. In this manner the I/O ports of the 8755A are bit-by-bit programmable as inputs or outputs. The table summarizes port and DDR designation. DDR's cannot be read.

AD <sub>1</sub>	AD <sub>0</sub>	Selection
0	0	Port A
0	1	Port B
1	0	Port A Data Direction Register (DDR A)
1	1	Port B Data Direction Register (DDR B)

When  $\overline{IOW}$  goes low and the Chip Enables are active, the data on the AD is written into I/O port selected by the latched value of AD<sub>0-1</sub>. During this operation all I/O bits of the selected port are affected, regardless of their I/O mode and the state of IO/M. The actual output level does not change until  $\overline{IOW}$  returns high. (glitch free output)

A port can be read out when the latched Chip Enables are active and either  $\overline{RD}$  goes low with IO/M high, or  $\overline{IOR}$  goes low. Both input and output mode bits of a selected port will appear on lines AD<sub>0-7</sub>.

To clarify the function of the I/O Ports and Data Direction Registers, the following diagram shows the configuration of one bit of PORT A and DDR A. The same logic applies to PORT B and DDR B.



Note that hardware RESET or writing a zero to the DDR latch will cause the output latch's output buffer to be disabled, preventing the data in the Output Latch from being passed through to the pin. This is equivalent to putting the port in the input mode. Note also that the data can be written to the Output Latch even though the Output Buffer has been disabled. This enables a port to be initialized with a value prior to enabling the output.

The diagram also shows that the contents of PORT A and PORT B can be read even when the ports are configured as outputs.

TABLE 1. 8755A PROGRAMMING MODULE CROSS REFERENCE

MODULE NAME	USE WITH
UPP 955	UPP(4)
UPP UP2(2)	UPP 855
PROMPT 975	PROMPT 80/85(3)
PROMPT 475	PROMPT 48(1)
NOTES:	
1. Described on p. 13-34 of 1978 Data Catalog.	
2. Special adaptor socket.	
3. Described on p. 13-39 of 1978 Data Catalog.	
4. Described on p. 13-71 of 1978 Data Catalog.	



The erasure characteristics of the 8755A are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8755A in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 8755A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8755 window to prevent unintentional erasure.

The recommended erasure procedure for the 8755A is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity X exposure time) for erasure should be a minimum of 15W-sec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000μW/cm<sup>2</sup> power rating. The 8755A should be placed within one inch from the lamp tubes during erasure. Some lamps have a filter on their tubes and this filter should be removed before erasure.

## PROGRAMMING

Initially, and after each erasure, all bits of the EPROM portions of the 8755A are in the "1" state. Information is introduced by selectively programming "0" into the desired bit locations. A programmed "0" can only be changed to a "1" by UV erasure.

The 8755A can be programmed on the Intel® Universal PROM Programmer (UPP), and the PROMPT™ 80/85 and PROMPT-48™ design aids. The appropriate programming modules and adapters for use in programming both 8755A's and 8755's are shown in Table 1.

The program mode itself consists of programming a single address at a time, giving a single 50 msec pulse for every address. Generally, it is desirable to have a verify cycle after a program cycle for the same address as shown in the attached timing diagram. In the verify cycle (i.e., normal memory read cycle) 'V<sub>DD</sub>' should be at +5V.

Preliminary timing diagrams and parameter values pertaining to the 8755A programming operation are contained in Figure 7.

## System Interface with 8085A and 8088

A system using the 8755A can use either one of the two I/O Interface techniques:

- Standard I/O
- Memory Mapped I/O

If a standard I/O technique is used, the system can use the feature of both CE<sub>3</sub> and CE<sub>1</sub>. By using a combination of unused address lines A<sub>11-15</sub> and the Chip Enable inputs, the 8085A system can use up to 5 each 8755A's without requiring a CE decoder. See Figure 2a and 2b.

If a memory mapped I/O approach is used the 8755A will be selected by the combination of both the Chip Enables and IO/M using the AD<sub>8-15</sub> address lines. See Figure 1.

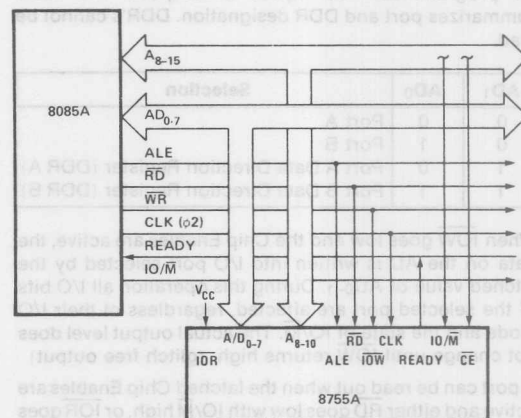
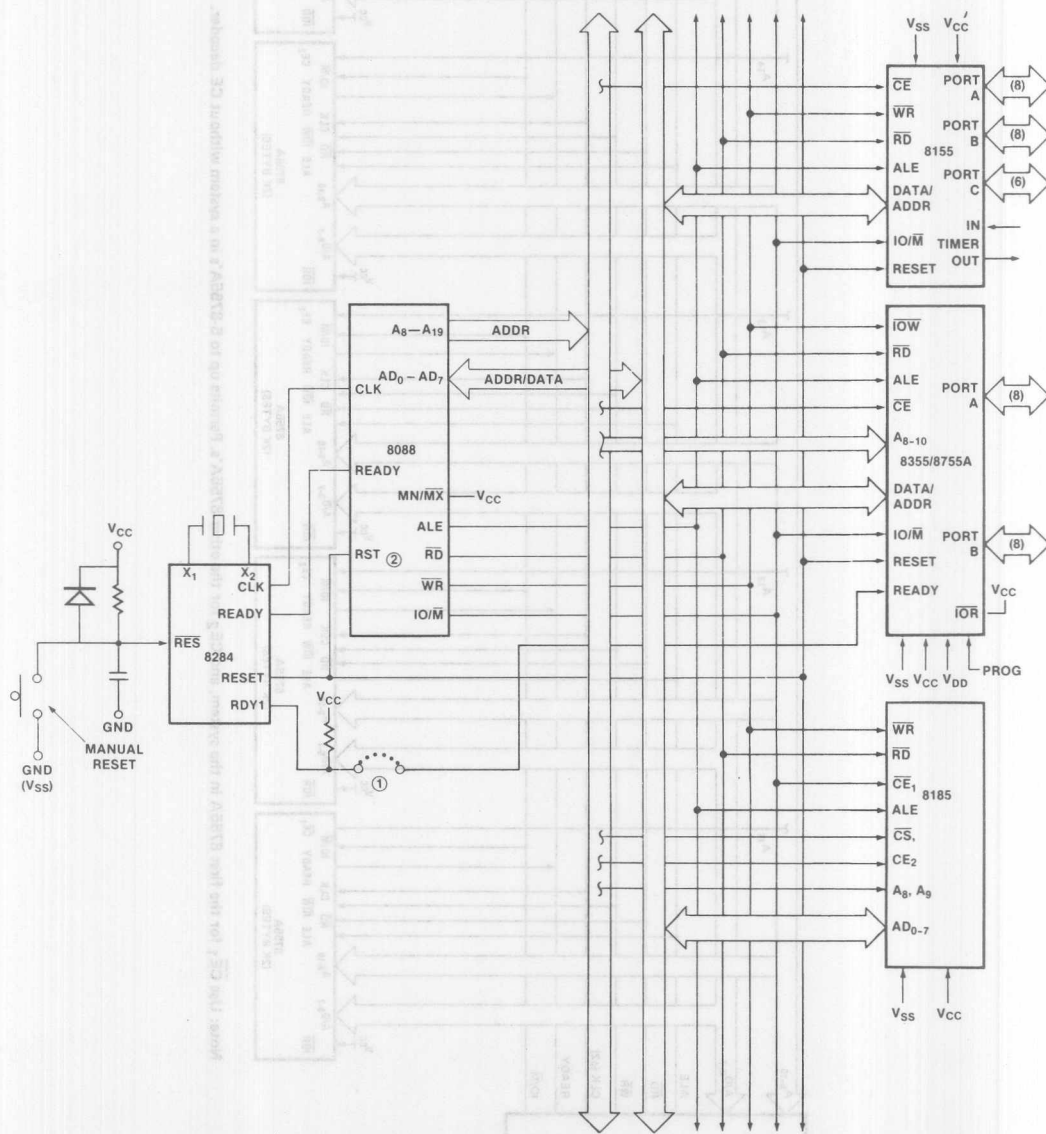


Figure 1. 8755A in 8085A System (Memory-Mapped I/O)

**8755A/8755A-2**

- 2 K Bytes ROM
- 38 I/O Pins
- 1 Interval Timer
- 2 Interrupt Levels



### Figure 2a. 8088 Five Chip System Configuration

6-50

**ABSOLUTE MAXIMUM RATINGS\***

Temperature Under Bias ..... 0°C to +70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage on Any Pin  
     With Respect to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 1.5W

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5\text{V} \pm 5\%$ )

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	$V_{CC} = 5.0\text{V}$
$V_{IH}$	Input High Voltage	2.0	$V_{CC} + 0.5$	V	$V_{CC} = 5.0\text{V}$
$V_{OL}$	Output Low Voltage		0.45	V	$I_{OL} = 2\text{mA}$
$V_{OH}$	Output High Voltage	2.4		V	$I_{OH} = -400\mu\text{A}$
$I_{IL}$	Input Leakage		10	$\mu\text{A}$	$V_{IN} = V_{CC}$ to 0V
$I_{LO}$	Output Leakage Current		$\pm 10$	$\mu\text{A}$	$0.45\text{V} \leq V_{OUT} \leq V_{CC}$
$I_{CC}$	$V_{CC}$ Supply Current		180	mA	

**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5\text{V} \pm 5\%$ )

Symbol	Parameter	8755A		8755A-2 (Preliminary)		Units
		Min.	Max.	Min.	Max.	
$t_{CYC}$	Clock Cycle Time	320		200		ns
$T_1$	CLK Pulse Width	80		40		ns
$T_2$	CLK Pulse Width	120		70		ns
$t_{r, tr}$	CLK Rise and Fall Time		30		30	ns
$t_{AL}$	Address to Latch Set Up Time	50		30		ns
$t_{LA}$	Address Hold Time after Latch	80		45		ns
$t_{LC}$	Latch to READ/WRITE Control	100		40		ns
$t_{RD}$	Valid Data Out Delay from READ Control		170		140	ns
$t_{AD}$	Address Stable to Data Out Valid		450		330	ns
$t_{LL}$	Latch Enable Width	100		70		ns
$t_{RDF}$	Data Bus Float after READ	0	100	0	85	ns
$t_{CL}$	READ/WRITE Control to Latch Enable	20		10		ns
$t_{CC}$	READ/WRITE Control Width	250		200		ns
$t_{DW}$	Data In to Write Set Up Time	150		150		ns
$t_{WD}$	Data In Hold Time After WRITE	30		10		ns
$t_{WP}$	WRITE to Port Output		400		400	ns
$t_{PR}$	Port Input Set Up Time	50		50		ns
$t_{RP}$	Port Input Hold Time	50		50		ns
$t_{RYH}$	READY HOLD Time	0	160	0	160	ns
$t_{ARY}$	ADDRESS (CE) to READY		160		160	ns
$t_{RV}$	Recovery Time Between Controls	300		200		ns
$t_{RDE}$	READ Control to Data Bus Enable	10		10		ns
$t_{LD}$	ALE to Data Out Valid		350		270	ns

Note:  $C_{LOAD} = 150\text{pF}$

## Input Waveform for A.C. Tests:



## WAVEFORMS

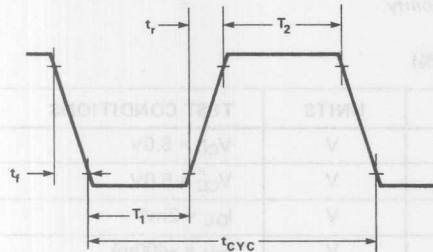


Figure 3. Clock Specification for 8755A

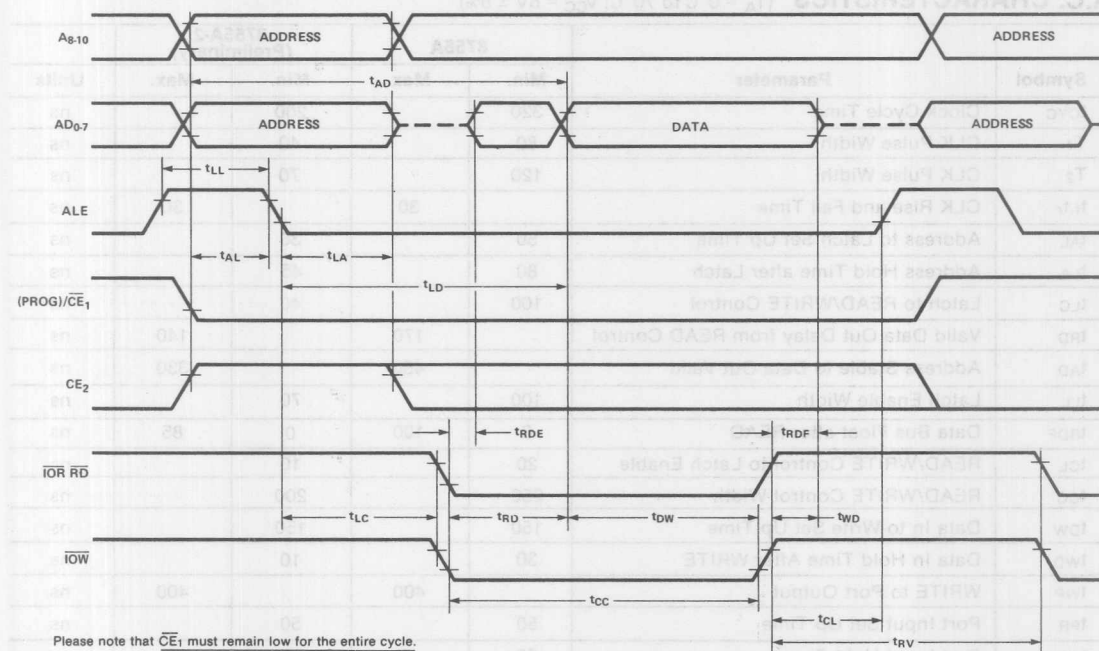


Figure 4. PROM Read, I/O Read and Write Timing



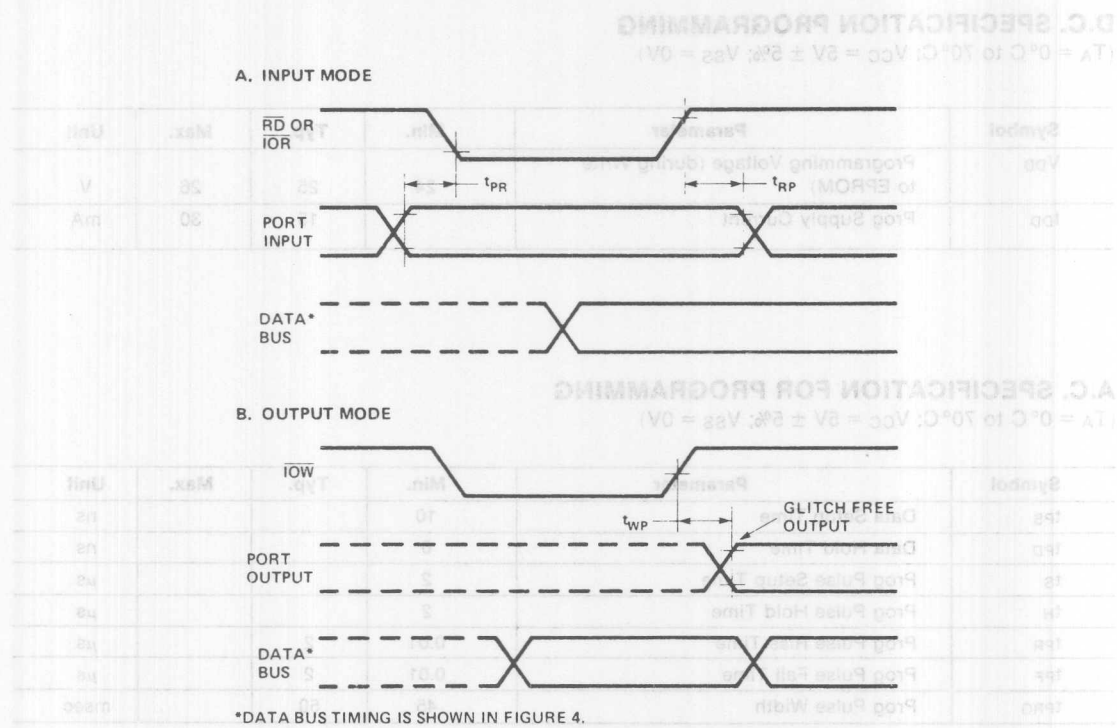


Figure 5. I/O Port Timing

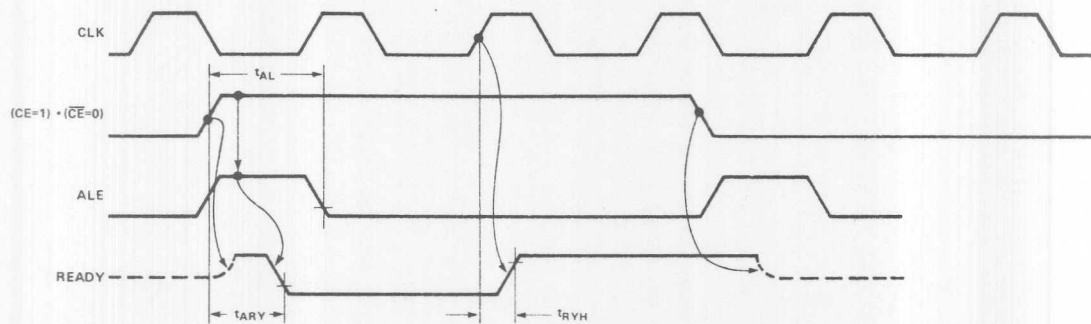


Figure 6. Wait State Timing (READY = 0)

Symbol	Parameter	Min.	Typ.	Max.	Unit
V <sub>DD</sub>	Programming Voltage (during Write to EPROM)	24	25	26	V
I <sub>DD</sub>	Prog Supply Current		15	30	mA

### A.C. SPECIFICATION FOR PROGRAMMING

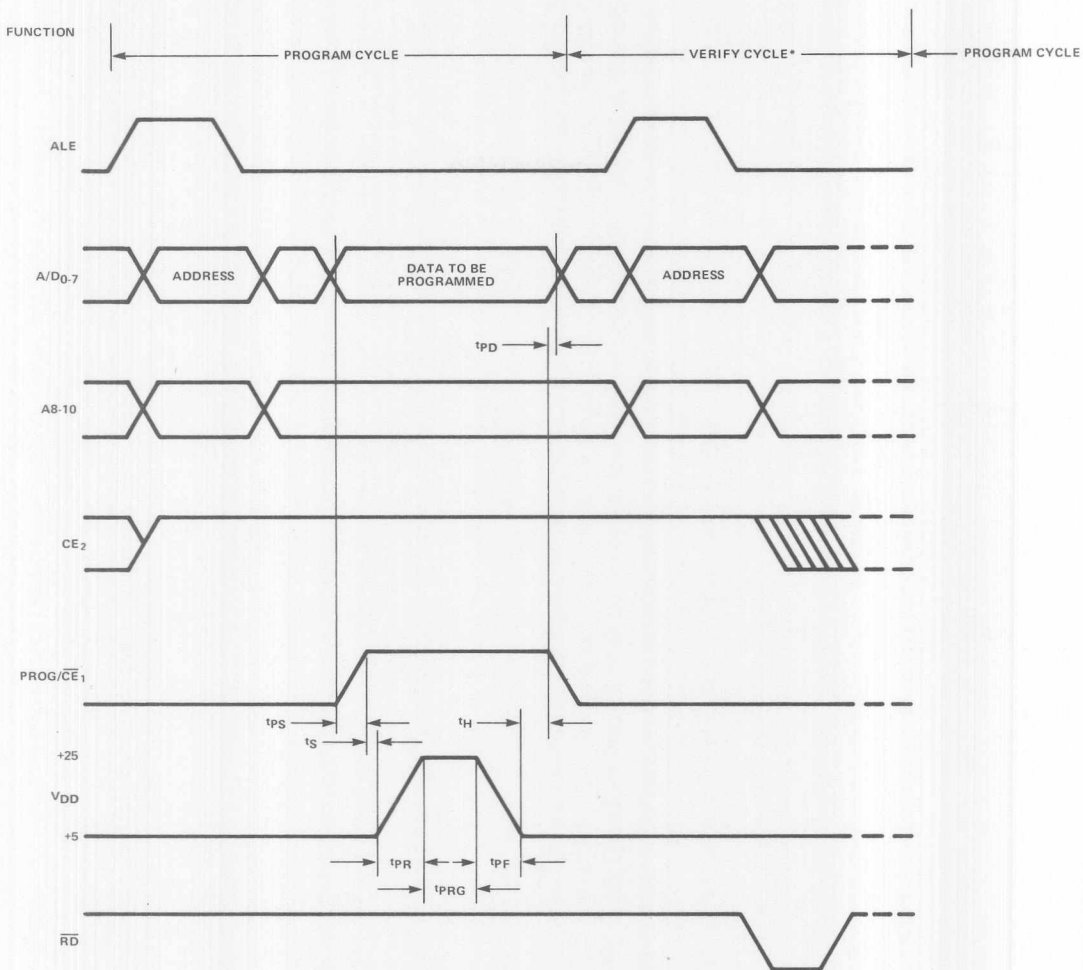
(T<sub>A</sub> = 0°C to 70°C; V<sub>CC</sub> = 5V ± 5%; V<sub>SS</sub> = 0V)

Symbol	Parameter	Min.	Typ.	Max.	Unit
t <sub>PS</sub>	Data Setup Time	10			ns
t <sub>PD</sub>	Data Hold Time	0			ns
t <sub>S</sub>	Prog Pulse Setup Time	2			μs
t <sub>H</sub>	Prog Pulse Hold Time	2			μs
t <sub>PR</sub>	Prog Pulse Rise Time	0.01	2		μs
t <sub>PF</sub>	Prog Pulse Fall Time	0.01	2		μs
t <sub>PRG</sub>	Prog Pulse Width	45	50		msec



Figure 6. Wait State Timing (READY = 0)

## WAVEFORMS



\*VERIFY CYCLE IS A REGULAR MEMORY READ CYCLE (WITH  $V_{DD} = +5V$  FOR 8755A)

Figure 7. 8755A Program Mode Timing Diagram



## Chapter 6

**MCS-85™**

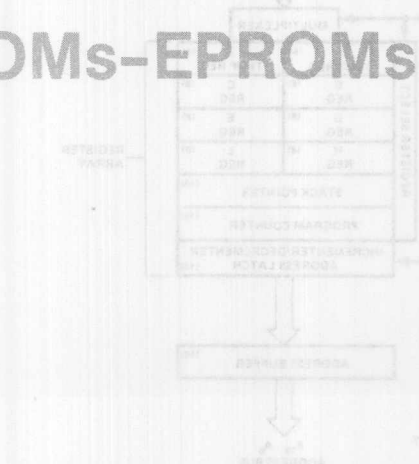
**MCS-80™**

# Systems Support Components

## Peripherals

## Static RAMs

## ROMs-EPROMs







## 8080A/8080A-1/8080A-2 8-BIT N-CANNEL MICROPROCESSOR

The 8080A is functionally and electrically compatible with the Intel® 8080.

- TTL Drive Capability
- 2  $\mu$ s (– 1:1.3  $\mu$ s, – 2:1.5  $\mu$ s) Instruction Cycle
- Powerful Problem Solving Instruction Set
- 6 General Purpose Registers and an Accumulator
- 16-Bit Program Counter for Directly Addressing up to 64K Bytes of Memory
- 16-Bit Stack Pointer and Stack Manipulation Instructions for Rapid Switching of the Program Environment
- Decimal, Binary, and Double Precision Arithmetic
- Ability to Provide Priority Vectored Interrupts
- 512 Directly Addressed I/O Ports

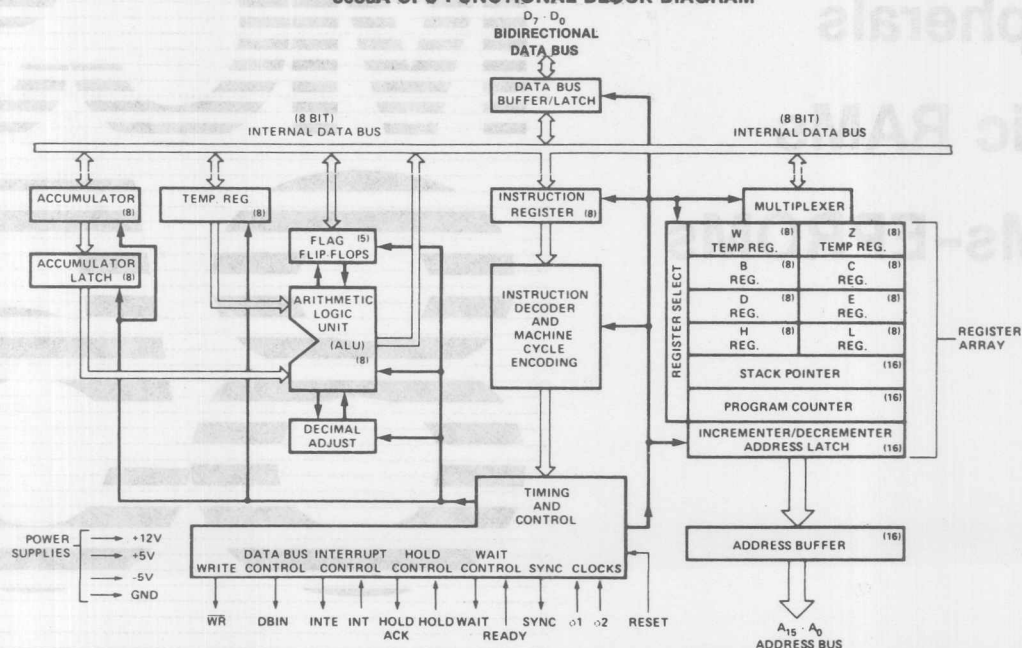
The Intel® 8080A is a complete 8-bit parallel central processing unit (CPU). It is fabricated on a single LSI chip using Intel's n-channel silicon gate MOS process. This offers the user a high performance solution to control and processing applications.

The 8080A contains 6 8-bit general purpose working registers and an accumulator. The 6 general purpose registers may be addressed individually or in pairs providing both single and double precision operators. Arithmetic and logical instructions set or reset 4 testable flags. A fifth flag provides decimal arithmetic operation.

The 8080A has an external stack feature wherein any portion of memory may be used as a last in/first out stack to store/retrieve the contents of the accumulator, flags, program counter, and all of the 6 general purpose registers. The 16-bit stack pointer controls the addressing of this external stack. This stack gives the 8080A the ability to easily handle multiple level priority interrupts by rapidly storing and restoring processor status. It also provides almost unlimited subroutine nesting.

This microprocessor has been designed to simplify systems design. Separate 16-line address and 8-line bidirectional data busses are used to facilitate easy interface to memory and I/O. Signals to control the interface to memory and I/O are provided directly by the 8080A. Ultimate control of the address and data busses resides with the HOLD signal. It provides the ability to suspend processor operation and force the address and data busses into a high impedance state. This permits OR-tying these busses with other controlling devices for (DMA) direct memory access or multi-processor operation.

8080A CPU FUNCTIONAL BLOCK DIAGRAM



## PIN DESCRIPTION

The following describes the function of all of the 8080A I/O pins. Several of the descriptions refer to internal timing periods.

### A<sub>15</sub>-A<sub>0</sub> (output three-state)

**ADDRESS BUS;** the address bus provides the address to memory (up to 64K 8-bit words) or denotes the I/O device number for up to 256 input and 256 output devices. A<sub>0</sub> is the least significant address bit.

### D<sub>7</sub>-D<sub>0</sub> (input/output three-state)

**DATA BUS;** the data bus provides bi-directional communication between the CPU, memory, and I/O devices for instructions and data transfers. Also, during the first clock cycle of each machine cycle, the 8080A outputs a status word on the data bus that describes the current machine cycle. D<sub>0</sub> is the least significant bit.

### SYNC (output)

**SYNCHRONIZING SIGNAL;** the SYNC pin provides a signal to indicate the beginning of each machine cycle.

### DBIN (output)

**DATA BUS IN;** the DBIN signal indicates to external circuits that the data bus is in the input mode. This signal should be used to enable the gating of data onto the 8080A data bus from memory or I/O.

### READY (input)

**READY;** the READY signal indicates to the 8080A that valid memory or input data is available on the 8080A data bus. This signal is used to synchronize the CPU with slower memory or I/O devices. If after sending an address out the 8080A does not receive a READY input, the 8080A will enter a WAIT state for as long as the READY line is low. READY can also be used to single step the CPU.

### WAIT (output)

**WAIT;** the WAIT signal acknowledges that the CPU is in a WAIT state.

### WR (output)

**WRITE;** the WR signal is used for memory WRITE or I/O output control. The data on the data bus is stable while the WR signal is active low ( $\overline{WR} = 0$ ).

### HOLD (input)

**HOLD;** the HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080A address and data bus as soon as the 8080A has completed its use of these buses for the current machine cycle. It is recognized under the following conditions:

- the CPU is in the HALT state.
  - the CPU is in the T<sub>2</sub> or T<sub>W</sub> state and the READY signal is active.
- As a result of entering the HOLD state the CPU ADDRESS BUS (A<sub>15</sub>-A<sub>0</sub>) and DATA BUS (D<sub>7</sub>-D<sub>0</sub>) will be in their high impedance state. The CPU acknowledges its state with the HOLD ACKNOWLEDGE (HLDA) pin.

### HLDA (output)

**HOLD ACKNOWLEDGE;** the HLDA signal appears in response to the HOLD signal and indicates that the data and address bus

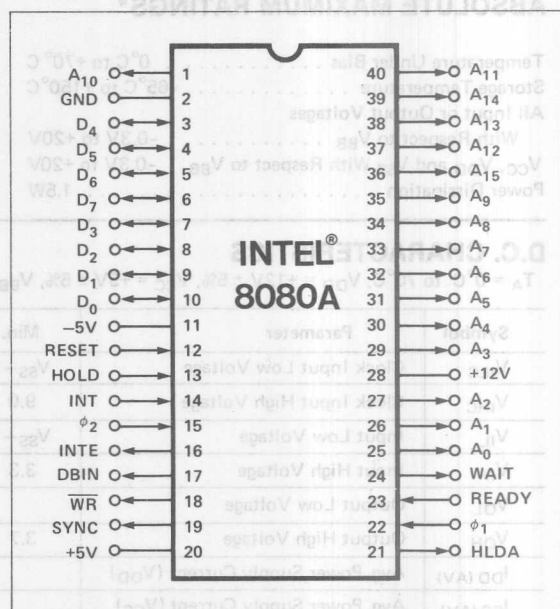


Figure 1. Pin Configuration

will go to the high impedance state. The HLDA signal begins at:

- T<sub>3</sub> for READ memory or input.
- The Clock Period following T<sub>3</sub> for WRITE memory or OUTPUT operation.

In either case, the HLDA signal appears after the rising edge of  $\phi_1$  and high impedance occurs after the rising edge of  $\phi_2$ .

### INTE (output)

**INTERRUPT ENABLE;** indicates the content of the internal interrupt enable flip/flop. This flip/flop may be set or reset by the Enable and Disable Interrupt instructions and inhibits interrupts from being accepted by the CPU when it is reset. It is automatically reset (disabling further interrupts) at time T<sub>1</sub> of the instruction fetch cycle (M1) when an interrupt is accepted and is also reset by the RESET signal.

### INT (input)

**INTERRUPT REQUEST;** the CPU recognizes an interrupt request on this line at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the Interrupt Enable flip/flop is reset it will not honor the request.

### RESET (input) [1]

**RESET;** while the RESET signal is activated, the content of the program counter is cleared. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip/flops are also reset. Note that the flags, accumulator, stack pointer, and registers are not cleared.

- V<sub>SS</sub> Ground Reference.
- V<sub>DD</sub> +12 ± 5% Volts.
- V<sub>CC</sub> +5 ± 5% Volts.
- V<sub>BB</sub> -5 ± 5% Volts (substrate bias).
- $\phi_1, \phi_2$  2 externally supplied clock phases. (non TTL compatible)

Temperature Under Bias	0°C to +70°C
Storage Temperature	-65°C to +150°C
All Input or Output Voltages	
With Respect to $V_{BB}$	-0.3V to +20V
$V_{CC}$ , $V_{DD}$ and $V_{SS}$ With Respect to $V_{BB}$	-0.3V to +20V
Power Dissipation	1.5W

"COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability."

## D.C. CHARACTERISTICS

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{DD} = +12\text{V} \pm 5\%$ ,  $V_{CC} = +5\text{V} \pm 5\%$ ,  $V_{BB} = -5\text{V} \pm 5\%$ ,  $V_{SS} = 0\text{V}$ , Unless Otherwise Noted.

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
$V_{ILC}$	Clock Input Low Voltage	$V_{SS}-1$		$V_{SS}+0.8$	V	
$V_{IHC}$	Clock Input High Voltage	9.0		$V_{DD}+1$	V	
$V_{IL}$	Input Low Voltage	$V_{SS}-1$		$V_{SS}+0.8$	V	
$V_{IH}$	Input High Voltage	3.3		$V_{CC}+1$	V	
$V_{OL}$	Output Low Voltage			0.45	V	$I_{OL} = 1.9\text{mA}$ on all outputs, $I_{OH} = -150\mu\text{A}$ .
$V_{OH}$	Output High Voltage	3.7			V	
$I_{DD}(AV)$	Avg. Power Supply Current ( $V_{DD}$ )		40	70	mA	Operation $T_{CY} = .48\mu\text{sec}$
$I_{CC}(AV)$	Avg. Power Supply Current ( $V_{CC}$ )		60	80	mA	
$I_{BB}(AV)$	Avg. Power Supply Current ( $V_{BB}$ )		.01	1	mA	
$I_{IL}$	Input Leakage			$\pm 10$	$\mu\text{A}$	$V_{SS} \leq V_{IN} \leq V_{CC}$
$I_{CL}$	Clock Leakage			$\pm 10$	$\mu\text{A}$	$V_{SS} \leq V_{CLOCK} \leq V_{DD}$
$I_{DL}^{[2]}$	Data Bus Leakage in Input Mode			-100 -2.0	$\mu\text{A}$ mA	$V_{SS} \leq V_{IN} \leq V_{SS} + 0.8\text{V}$ $V_{SS} + 0.8\text{V} \leq V_{IN} \leq V_{CC}$
$I_{FL}$	Address and Data Bus Leakage During HOLD			+10 -100	$\mu\text{A}$	$V_{ADDR/DATA} = V_{CC}$ $V_{ADDR/DATA} = V_{SS} + 0.45\text{V}$

## CAPACITANCE

$T_A = 25^\circ\text{C}$   $V_{CC} = V_{DD} = V_{SS} = 0\text{V}$ ,  $V_{BB} = -5\text{V}$

Symbol	Parameter	Typ.	Max.	Unit	Test Condition
$C_\phi$	Clock Capacitance	17	25	pf	$f_c = 1\text{MHz}$
$C_{IN}$	Input Capacitance	6	10	pf	Unmeasured Pins
$C_{OUT}$	Output Capacitance	10	20	pf	Returned to $V_{SS}$

### NOTES:

1. The RESET signal must be active for a minimum of 3 clock cycles.
2. When DBIN is high and  $V_{IN} > V_{IH}$  an internal active pull up will be switched onto the Data Bus.
3.  $\Delta I_{supply} / \Delta T_A = -0.45\%/^\circ\text{C}$ .

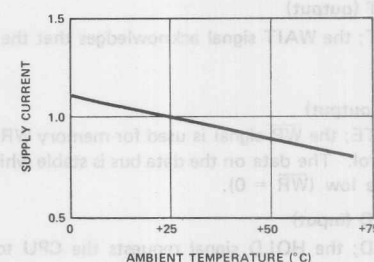


Figure 2. Typical Supply Current vs. Temperature, Normalized<sup>[3]</sup>

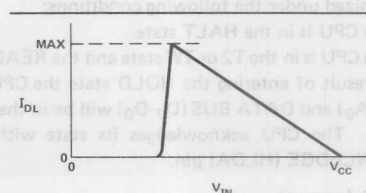


Figure 3. Data Bus Characteristic During DBIN

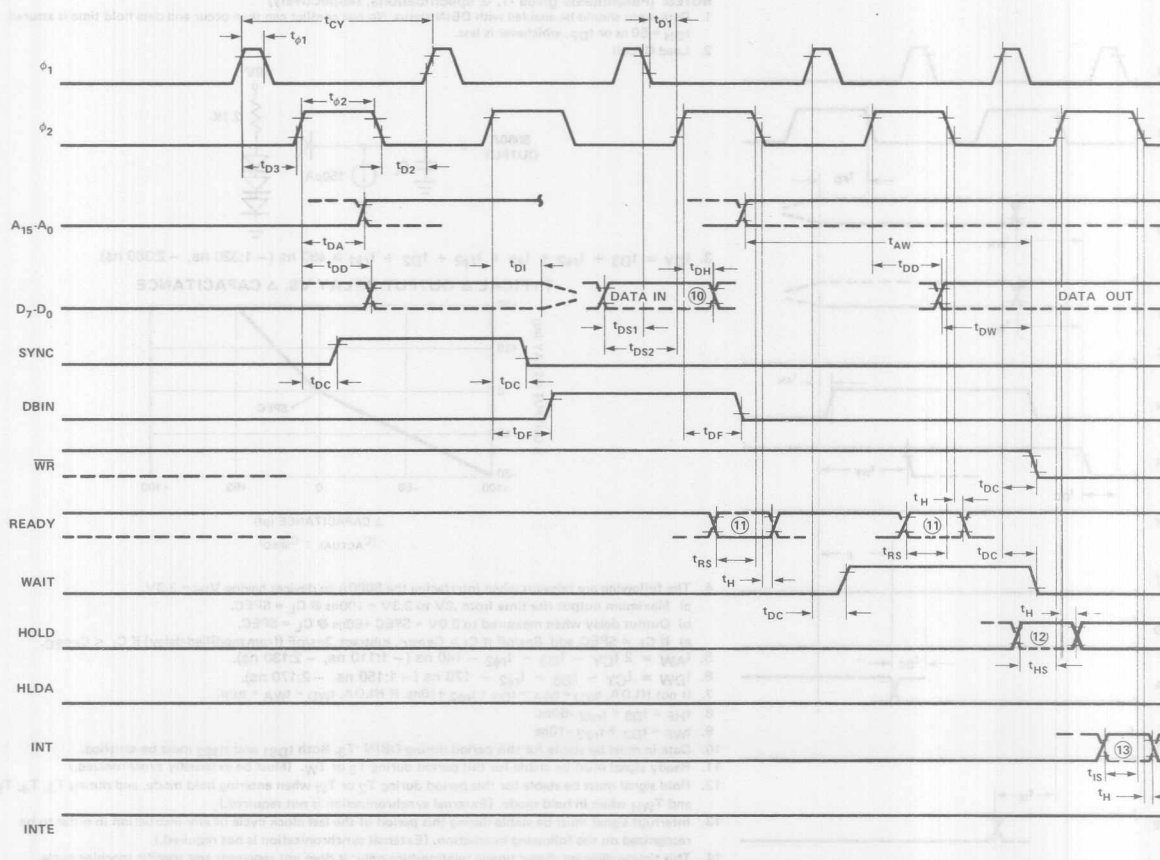
## A.C. CHARACTERISTICS (8080A)

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{DD} = +12\text{V} \pm 5\%$ ,  $V_{CC} = +5\text{V} \pm 5\%$ ,  $V_{BB} = -5\text{V} \pm 5\%$ ,  $V_{SS} = 0\text{V}$ , Unless Otherwise Noted

Symbol	Parameter	Min.	Max.	-1 Min.	-1 Max.	-2 Min.	-2 Max.	Unit	Test Condition
$t_{CY}^{[3]}$	Clock Period	0.48	2.0	0.32	2.0	0.38	2.0	$\mu\text{sec}$	
$t_r, t_f$	Clock Rise and Fall Time	0	50	0	25	0	50	nsec	
$t_{\phi 1}$	$\phi_1$ Pulse Width	60		50		60		nsec	
$t_{\phi 2}$	$\phi_2$ Pulse Width	220		145		175		nsec	
$t_{D1}$	Delay $\phi_1$ to $\phi_2$	0		0		0		nsec	
$t_{D2}$	Delay $\phi_2$ to $\phi_1$	70		60		70		nsec	
$t_{D3}$	Delay $\phi_1$ to $\phi_2$ Leading Edges	80		60		70		nsec	
$t_{DA}^{[2]}$	Address Output Delay From $\phi_2$		200		150		175	nsec	$C_L = 100\text{ pF}$
$t_{DD}^{[2]}$	Data Output Delay From $\phi_2$		220		180		200	nsec	
$t_{DC}^{[2]}$	Signal Output Delay From $\phi_2$ or $\phi_1$ (SYNC, WR, WAIT, HLDA)		120		110		120	nsec	$C_L = 50\text{ pF}$
$t_{DF}^{[2]}$	DBIN Delay From $\phi_2$		25	140	25	130	25	140	nsec
$t_{DI}^{[1]}$	Delay for Input Bus to Enter Input Mode		$t_{DF}$		$t_{DF}$		$t_{DF}$	nsec	
$t_{DS1}$	Data Setup Time During $\phi_1$ and DBIN	30		10		20		nsec	

## WAVEFORMS

(Note: Timing measurements are made at the following reference voltages: CLOCK "1" = 8.0V "0" = 1.0V; INPUTS "1" = 3.3V, "0" = 0.8V; OUTPUTS "1" = 2.0V, "0" = 0.8V.)



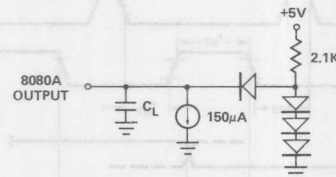
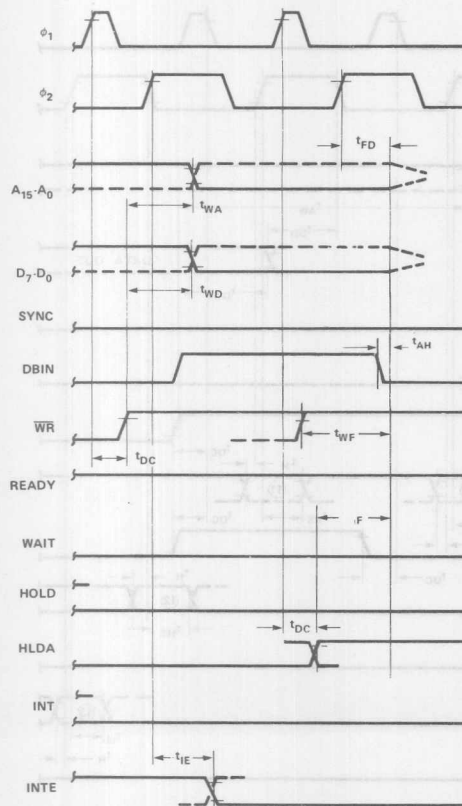
## A.C. CHARACTERISTICS (8080A)

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{DD} = +12\text{V} \pm 5\%$ ,  $V_{CC} = +5\text{V} \pm 5\%$ ,  $V_{BB} = -5\text{V} \pm 5\%$ ,  $V_{SS} = 0\text{V}$ , Unless Otherwise Noted

Symbol	Parameter	Min.	Max.	-1 Min.	-1 Max.	-2 Min.	-2 Max.	Unit	Test Condition
$t_{DS2}$	Data Setup Time to $\phi_2$ During DBIN	150		120		130		nsec	$C_L = 50\text{ pF}$
$t_{DH}^{[1]}$	Data Hold time From $\phi_2$ During DBIN	[1]		[1]		[1]		nsec	
$t_{IE}^{[2]}$	INTE Output Delay From $\phi_2$		200		200		200	nsec	
$t_{RS}$	READY Setup Time During $\phi_2$	120		90		90		nsec	
$t_{HS}$	HOLD Setup Time to $\phi_2$	140		120		120		nsec	
$t_{IS}$	INT Setup Time During $\phi_2$	120		100		100		nsec	$C_L = 100\text{ pF}$ : Address, Data $C_L = 50\text{ pF}$ : WR, HLDA, DBIN
$t_H$	Hold Time From $\phi_2$ (READY, INT, HOLD)	0		0		0		nsec	
$t_{FD}$	Delay to Float During Hold (Address and Data Bus)		120		120		120	nsec	
$t_{AW}^{[2]}$	Address Stable Prior to WR	[5]		[5]		[5]		nsec	
$t_{DW}^{[2]}$	Output Data Stable Prior to WR	[6]		[6]		[6]		nsec	
$t_{WD}^{[2]}$	Output Data Stable From WR	[7]		[7]		[7]		nsec	
$t_{WA}^{[2]}$	Address Stable From WR	[7]		[7]		[7]		nsec	
$t_{HF}^{[2]}$	HLDA to Float Delay	[8]		[8]		[8]		nsec	
$t_{WF}^{[2]}$	WR to Float Delay	[9]		[9]		[9]		nsec	
$t_{AH}^{[2]}$	Address Hold Time After DBIN During HLDA	-20		-20		-20		nsec	

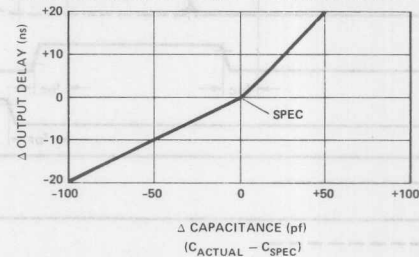
NOTES: (Parenthesis gives -1, -2 specifications, respectively)

1. Data input should be enabled with DBIN status. No bus conflict can then occur and data hold time is assured.  $t_{DH} = 50\text{ ns}$  or  $t_{DF}$ , whichever is less.
2. Load Circuit.



$$3. t_{CY} = t_{D3} + t_{r\phi2} + t_{\phi2} + t_{f\phi2} + t_{D2} + t_{r\phi1} \geq 480\text{ ns} \quad (-1:320\text{ ns}, -2:380\text{ ns}).$$

TYPICAL  $\Delta$  OUTPUT DELAY VS.  $\Delta$  CAPACITANCE



4. The following are relevant when interfacing the 8080A to devices having  $V_{IH} = 3.3\text{V}$ :
  - a) Maximum output rise time from .8V to 3.3V = 100ns @  $C_L = \text{SPEC}$ .
  - b) Output delay when measured to 3.0V = SPEC + 60ns @  $C_L = \text{SPEC}$ .
  - c) If  $C_L \neq \text{SPEC}$ , add .6ns/pF if  $C_L > C_{\text{SPEC}}$ , subtract .3ns/pF (from modified delay) if  $C_L < C_{\text{SPEC}}$ .
5.  $t_{AW} = 2 t_{CY} - t_{D3} - t_{r\phi2} - 140\text{ ns}$  (-1:110 ns, -2:130 ns).
6.  $t_{DW} = t_{CY} - t_{D3} - t_{r\phi2} - 170\text{ ns}$  (-1:150 ns, -2:170 ns).
7. If not HLDA,  $t_{WD} = t_{WA} = t_{D3} + t_{r\phi2} + 10\text{ns}$ . If HLDA,  $t_{WD} = t_{WA} = t_{WF}$ .
8.  $t_{HF} = t_{D3} + t_{r\phi2} - 50\text{ns}$ .
9.  $t_{WF} = t_{D3} + t_{r\phi2} - 10\text{ns}$ .
10. Data in must be stable for this period during DBIN  $\cdot T_3$ . Both  $t_{DS1}$  and  $t_{DS2}$  must be satisfied.
11. Ready signal must be stable for this period during  $T_2$  or  $T_W$ . (Must be externally synchronized.)
12. Hold signal must be stable for this period during  $T_2$  or  $T_W$  when entering hold mode, and during  $T_3$ ,  $T_4$ ,  $T_5$  and  $T_{WH}$  when in hold mode. (External synchronization is not required.)
13. Interrupt signal must be stable during this period of the last clock cycle of any instruction in order to be recognized on the following instruction. (External synchronization is not required.)
14. This timing diagram shows timing relationships only; it does not represent any specific machine cycle.



## INSTRUCTION SET

The accumulator group instructions include arithmetic and logical operators with direct, indirect, and immediate addressing modes.

Move, load, and store instruction groups provide the ability to move either 8 or 16 bits of data between memory, the six working registers and the accumulator using direct, indirect, and immediate addressing modes.

The ability to branch to different portions of the program is provided with jump, jump conditional, and computed jumps. Also the ability to call to and return from sub-routines is provided both conditionally and unconditionally. The RESTART (or single byte call instruction) is useful for interrupt vector operation.

Double precision operators such as stack manipulation and double add instructions extend both the arithmetic and interrupt handling capability of the 8080A. The ability to

increment and decrement memory, the six general registers and the accumulator is provided as well as extended increment and decrement instructions to operate on the register pairs and stack pointer. Further capability is provided by the ability to rotate the accumulator left or right through or around the carry bit.

Input and output may be accomplished using memory addresses as I/O ports or the directly addressed I/O provided for in the 8080A instruction set.

The following special instruction group completes the 8080A instruction set: the NOP instruction, HALT to stop processor execution and the DAA instructions provide decimal arithmetic capability. STC allows the carry flag to be directly set, and the CMC instruction allows it to be complemented. CMA complements the contents of the accumulator and XCHG exchanges the contents of two 16-bit register pairs directly.

### Data and Instruction Formats

Data in the 8080A is stored in the form of 8-bit binary integers. All data transfers to the system data bus will be in the same format.

D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub>

#### DATA WORD

The program instructions may be one, two, or three bytes in length. Multiple byte instructions must be stored in successive bytes in program memory. The instruction formats then depend on the particular operation executed.

#### One Byte Instructions

D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub> OP CODE

#### Two Byte Instructions

D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub> OP CODE

D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub> OPERAND

#### Three Byte Instructions

D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub> OP CODE

D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub> LOW ADDRESS OR OPERAND 1

D<sub>7</sub> D<sub>6</sub> D<sub>5</sub> D<sub>4</sub> D<sub>3</sub> D<sub>2</sub> D<sub>1</sub> D<sub>0</sub> HIGH ADDRESS OR OPERAND 2

For the 8080A a logic "1" is defined as a high level and a logic "0" is defined as a low level.

### TYPICAL INSTRUCTIONS

Register to register, memory reference, arithmetic or logical, rotate, return, push, pop, enable or disable  
Interrupt instructions

Immediate mode or I/O instructions

Jump, call or direct load and store instructions

## 8080 INSTRUCTION SET

## Summary of Processor Instructions

Mnemonic	Description	Instruction Code[1]								Clock[2]
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	
MOVE, LOAD, AND STORE										
MOV r,r2	Move register to register	0	1	D	D	D	S	S	S	5
MOV M,r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r,M	Move memory to register	0	1	D	D	D	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
LXI B	Load immediate register	0	0	0	0	0	0	0	1	10
	Pair B & C									
LXI D	Load immediate register	0	0	0	1	0	0	0	1	10
	Pair D & E									
LXI H	Load immediate register	0	0	1	0	0	0	0	1	10
	Pair H & L									
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
STACK OPS										
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POP B	Pop register Pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register Pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register Pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
JUMP										
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
INSTRUCTION SET										
Mnemonic	Description	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	Cycles
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
CALL										
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RETURN										
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RESTART										
RST	Restart	1	1	A	A	A	1	1	1	11
INCREMENT AND DECREMENT										
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
ADD										
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10

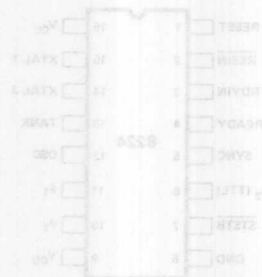
NOTES: 1. DDD or SSS, B 000, C 001, D 010, E 011, H 100, L 101, Memory 110, A 111.

2. Two possible cycle times. (6/12) indicate instruction cycles dependent on condition flags.

\*All mnemonics copyright  
© Intel Corporation 1977

## Summary of Processor Instructions (Cont.)

Mnemonic	Description	Instruction Code[1]								Clock[2]
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	
SUBTRACT										
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
LOGICAL										
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
ROTATE										
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
SPECIALS										
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
INPUT/OUTPUT										
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
CONTROL										
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable Interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4
HLT	Halt	0	1	1	1	0	1	1	0	7



Pin	Signal	Function
1	VCC	Power supply
2	A15	Address bus
3	A14	Address bus
4	A13	Address bus
5	A12	Address bus
6	A11	Address bus
7	A10	Address bus
8	A9	Address bus
9	A8	Address bus
10	A7	Address bus
11	A6	Address bus
12	A5	Address bus
13	A4	Address bus
14	A3	Address bus
15	A2	Address bus
16	A1	Address bus
17	A0	Address bus
18	VCC	Power supply
19	B7	Data bus
20	B6	Data bus
21	B5	Data bus
22	B4	Data bus
23	B3	Data bus
24	B2	Data bus
25	B1	Data bus
26	B0	Data bus
27	VCC	Power supply
28	CLOCK	Clock signal
29	READY	Ready signal
30	RD	Read control signal
31	WR	Write control signal
32	A16	Address bus
33	A17	Address bus
34	A18	Address bus
35	A19	Address bus
36	A20	Address bus
37	A21	Address bus
38	A22	Address bus
39	A23	Address bus
40	VCC	Power supply

NOTES: 1. DDD or SSS: B=000, C=001, D=010, E=011, H=100, L=101. Memory=110, A=111.

2. Two possible cycle times. (6/12) indicate instruction cycles dependent on condition flags.

\*All mnemonics copyright

© Intel Corporation 1977

# 8224

## CLOCK GENERATOR AND DRIVER FOR 8080A CPU

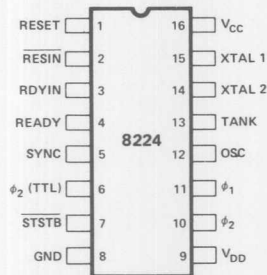
- Single Chip Clock Generator/Driver for 8080A CPU
- Power-Up Reset for CPU
- Ready Synchronizing Flip-Flop
- Advanced Status Strobe
- Oscillator Output for External System Timing
- Crystal Controlled for Stable System Operation
- Reduces System Package Count

The Intel® 8224 is a single chip clock generator/driver for the 8080A CPU. It is controlled by a crystal, selected by the designer to meet a variety of system speed requirements.

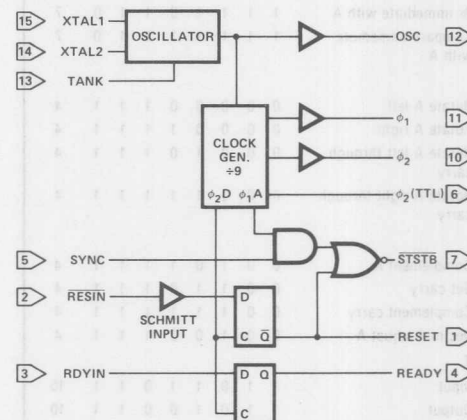
Also included are circuits to provide power-up reset, advance status strobe, and synchronization of ready.

The 8224 provides the designer with a significant reduction of packages used to generate clocks and timing for 8080A.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

RESIN	RESET INPUT
RESET	RESET OUTPUT
RDYIN	READY INPUT
READY	READY OUTPUT
SYNC	SYNC INPUT
STSTB	STATUS STB (ACTIVE LOW)
phi1	8080
phi2	CLOCKS

XTAL 1	CONNECTIONS FOR CRYSTAL
XTAL 2	
TANK	USED WITH OVERTONE XTAL
OSC	OSCILLATOR OUTPUT
phi2 (TTL)	phi2 CLK (TTL LEVEL)
VCC	+5V
VDD	+12V
GND	0V

**ABSOLUTE MAXIMUM RATINGS\***

Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to 150°C
Supply Voltage, $V_{CC}$	-0.5V to +7V
Supply Voltage, $V_{DD}$	-0.5V to +13.5V
Input Voltage	-1.5V to +7V
Output Current	100mA

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS**

$T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ;  $V_{CC} = +5.0\text{V} \pm 5\%$ ;  $V_{DD} = +12\text{V} \pm 5\%$ .

Symbol	Parameter	Limits			Units	Test Conditions
		Min.	Typ.	Max.		
$I_F$	Input Current Loading			-.25	mA	$V_F = .45\text{V}$
$I_R$	Input Leakage Current			10	$\mu\text{A}$	$V_R = 5.25\text{V}$
$V_C$	Input Forward Clamp Voltage			1.0	V	$I_C = -5\text{mA}$
$V_{IL}$	Input "Low" Voltage			.8	V	$V_{CC} = 5.0\text{V}$
$V_{IH}$	Input "High" Voltage	2.6 2.0			V	Reset Input All Other Inputs
$V_{IH}-V_{IL}$	RESIN Input Hysteresis	.25			V	$V_{CC} = 5.0\text{V}$
$V_{OL}$	Output "Low" Voltage			.45 .45	V V	$(\phi_1, \phi_2)$ , Ready, Reset, $\overline{\text{STSTB}}$ $I_{OL} = 2.5\text{mA}$ All Other Outputs $I_{OL} = 15\text{mA}$
$V_{OH}$	Output "High" Voltage					
	$\phi_1, \phi_2$	9.4			V	$I_{OH} = -100\mu\text{A}$
	READY, RESET	3.6			V	$I_{OH} = -100\mu\text{A}$
	All Other Outputs	2.4			V	$I_{OH} = -1\text{mA}$
$I_{SC}^{[1]}$	Output Short Circuit Current (All Low Voltage Outputs Only)	-10		-60	mA	$V_O = 0\text{V}$ $V_{CC} = 5.0\text{V}$
$I_{CC}$	Power Supply Current			115	mA	
$I_{DD}$	Power Supply Current			12	mA	

Note: 1. Caution,  $\phi_1$  and  $\phi_2$  output drivers do not have short circuit protection

**Crystal Requirements**

Tolerance: 0.005% at 0°C-70°C  
 Resonance: Series (Fundamental)\*  
 Load Capacitance: 20-35 pF  
 Equivalent Resistance: 75-20 ohms  
 Power Dissipation (Min): 4 mW

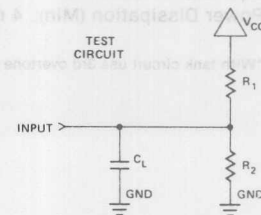
\*With tank circuit use 3rd overtone mode.



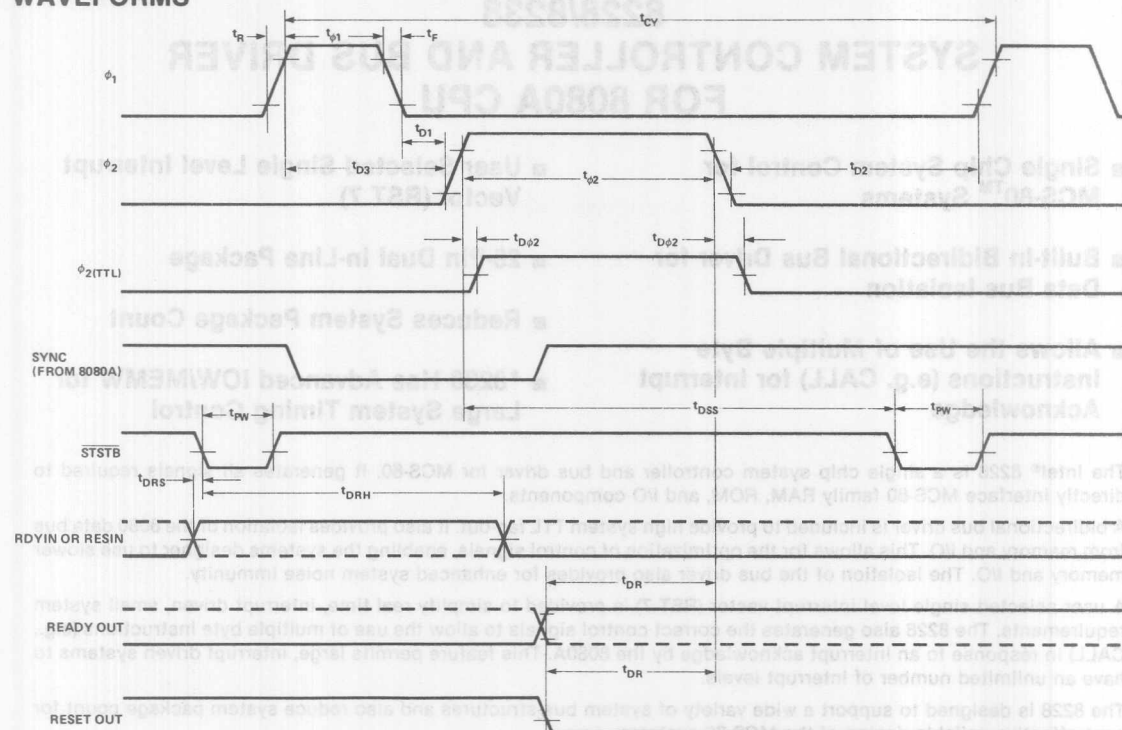
## A.C. CHARACTERISTICS

$V_{CC} = +5.0V \pm 5\%$ ;  $V_{DD} = +12.0V \pm 5\%$ ;  $T_A = 0^\circ C$  to  $70^\circ C$

Symbol	Parameter	Limits			Units	Test Conditions
		Min.	Typ.	Max.		
$t_{\phi 1}$	$\phi_1$ Pulse Width	$\frac{2t_{cy}}{9} - 20ns$				
$t_{\phi 2}$	$\phi_2$ Pulse Width	$\frac{5t_{cy}}{9} - 35ns$				
$t_{D1}$	$\phi_1$ to $\phi_2$ Delay	0			ns	
$t_{D2}$	$\phi_2$ to $\phi_1$ Delay	$\frac{2t_{cy}}{9} - 14ns$				$C_L = 20pF$ to $50pF$
$t_{D3}$	$\phi_1$ to $\phi_2$ Delay	$\frac{2t_{cy}}{9}$		$\frac{2t_{cy}}{9} + 20ns$		
$t_R$	$\phi_1$ and $\phi_2$ Rise Time			20		
$t_F$	$\phi_1$ and $\phi_2$ Fall Time			20		
$t_{D\phi 2}$	$\phi_2$ to $\phi_2$ (TTL) Delay	-5		+15	ns	$\phi_2$ TTL, $C_L=30$ $R_1=300\Omega$ $R_2=600\Omega$
$t_{DSS}$	$\phi_2$ to $\overline{STSTB}$ Delay	$\frac{6t_{cy}}{9} - 30ns$		$\frac{6t_{cy}}{9}$		
$t_{PW}$	$\overline{STSTB}$ Pulse Width	$\frac{t_{cy}}{9} - 15ns$				$\overline{STSTB}$ , $C_L=15pF$ $R_1 = 2K$ $R_2 = 4K$
$t_{DRS}$	RDYIN Setup Time to Status Strobe	$50ns - \frac{4t_{cy}}{9}$				
$t_{DRH}$	RDYIN Hold Time After $\overline{STSTB}$	$\frac{4t_{cy}}{9}$				
$t_{DR}$	RDYIN or RESIN to $\phi_2$ Delay	$\frac{4t_{cy}}{9} - 25ns$				Ready & Reset $C_L=10pF$ $R_1=2K$ $R_2=4K$
$t_{CLK}$	CLK Period		$\frac{t_{cy}}{9}$			
$f_{max}$	Maximum Oscillating Frequency			27	MHz	
$C_{in}$	Input Capacitance			8	pF	$V_{CC}=+5.0V$ $V_{DD}=+12V$ $V_{BIAS}=2.5V$ $f=1MHz$



## WAVEFORMS



VOLTAGE MEASUREMENT POINTS:  $\phi_1, \phi_2$  Logic "0" = 1.0V, Logic "1" = 8.0V. All other signals measured at 1.5V.

## EXAMPLE:

A.C. CHARACTERISTICS (For  $t_{CY} = 488.28 \text{ ns}$ )

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{DD} = +5\text{V} \pm 5\%$ ;  $V_{DD} = +12\text{V} \pm 5\%$ .

Symbol	Parameter	Limits			Units	Test Conditions
		Min.	Typ.	Max.		
$t_{\phi 1}$	$\phi_1$ Pulse Width	89			ns	$t_{CY} = 488.28 \text{ ns}$  $\phi_1$ & $\phi_2$ Loaded to $C_L = 20$ to $50 \text{ pF}$
$t_{\phi 2}$	$\phi_2$ Pulse Width	236			ns	
$t_{D1}$	Delay $\phi_1$ to $\phi_2$	0			ns	
$t_{D2}$	Delay $\phi_2$ to $\phi_1$	95			ns	
$t_{D3}$	Delay $\phi_1$ to $\phi_2$ Leading Edges	109		129	ns	
$t_r$	Output Rise Time			20	ns	
$t_f$	Output Fall Time			20	ns	Ready & Reset Loaded to $2 \text{ mA}/10 \text{ pF}$ All measurements referenced to 1.5V unless specified otherwise.
$t_{DSS}$	$\phi_2$ to $\overline{\text{STSTB}}$ Delay	296		326	ns	
$t_{D\phi 2}$	$\phi_2$ to $\phi_2$ (TTL) Delay	-5		+15	ns	
$t_{PW}$	Status Strobe Pulse Width	40			ns	
$t_{DRS}$	$\text{RDYIN}$ Setup Time to $\overline{\text{STSTB}}$	-167			ns	
$t_{DRH}$	$\text{RDYIN}$ Hold Time after $\overline{\text{STSTB}}$	217			ns	
$t_{DR}$	$\text{READY}$ or $\text{RESET}$ to $\phi_2$ Delay	192			ns	
$f_{MAX}$	Oscillator Frequency			18.432	MHz	



## 8228/8238 SYSTEM CONTROLLER AND BUS DRIVER FOR 8080A CPU

- Single Chip System Control for MCS-80™ Systems
- Built-In Bidirectional Bus Driver for Data Bus Isolation
- Allows the Use of Multiple Byte Instructions (e.g. CALL) for Interrupt Acknowledge
- User Selected Single Level Interrupt Vector (RST 7)
- 28-Pin Dual In-Line Package
- Reduces System Package Count
- \*8238 Has Advanced IOW/MEMW for Large System Timing Control

The Intel® 8228 is a single chip system controller and bus driver for MCS-80. It generates all signals required to directly interface MCS-80 family RAM, ROM, and I/O components.

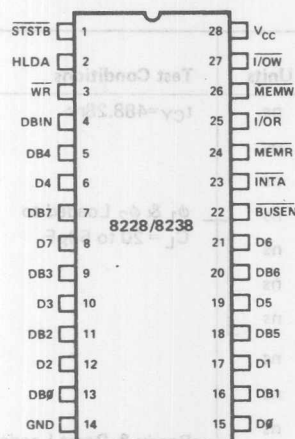
A bidirectional bus driver is included to provide high system TTL fan-out. It also provides isolation of the 8080 data bus from memory and I/O. This allows for the optimization of control signals, enabling the systems designer to use slower memory and I/O. The isolation of the bus driver also provides for enhanced system noise immunity.

A user selected single level interrupt vector (RST 7) is provided to simplify real time, interrupt driven, small system requirements. The 8228 also generates the correct control signals to allow the use of multiple byte instructions (e.g., CALL) in response to an interrupt acknowledge by the 8080A. This feature permits large, interrupt driven systems to have an unlimited number of interrupt levels.

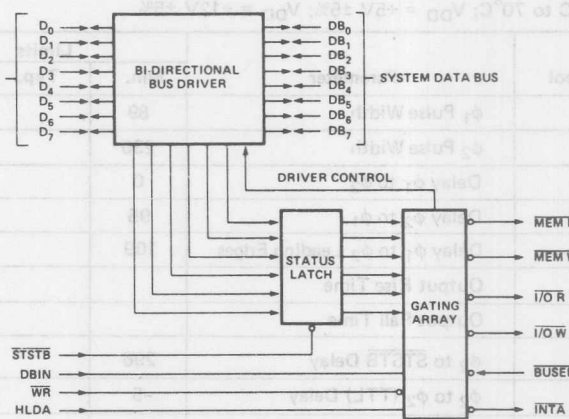
The 8228 is designed to support a wide variety of system bus structures and also reduce system package count for cost effective, reliable design of the MCS-80 systems.

Note: The specifications for the 3228/3238 are identical with those for the 8228/8238

### PIN CONFIGURATION



### BLOCK DIAGRAM



### PIN NAMES

D7-D0	DATA BUS (8080 SIDE)	INTA	INTERRUPT ACKNOWLEDGE
DB7-DB0	DATA BUS (SYSTEM SIDE)	HLDA	HLDA (FROM 8080)
I/OR	I/O READ	WR	WR (FROM 8080)
I/OW	I/O WRITE	BUSEN	BUS ENABLE INPUT
MEMR	MEMORY READ	STSB	STATUS STROBE (FROM 8224)
MEMW	MEMORY WRITE	Vcc	+5V
DBIN	DBIN (FROM 8080)	GND	0 VOLTS

**ABSOLUTE MAXIMUM RATINGS\***

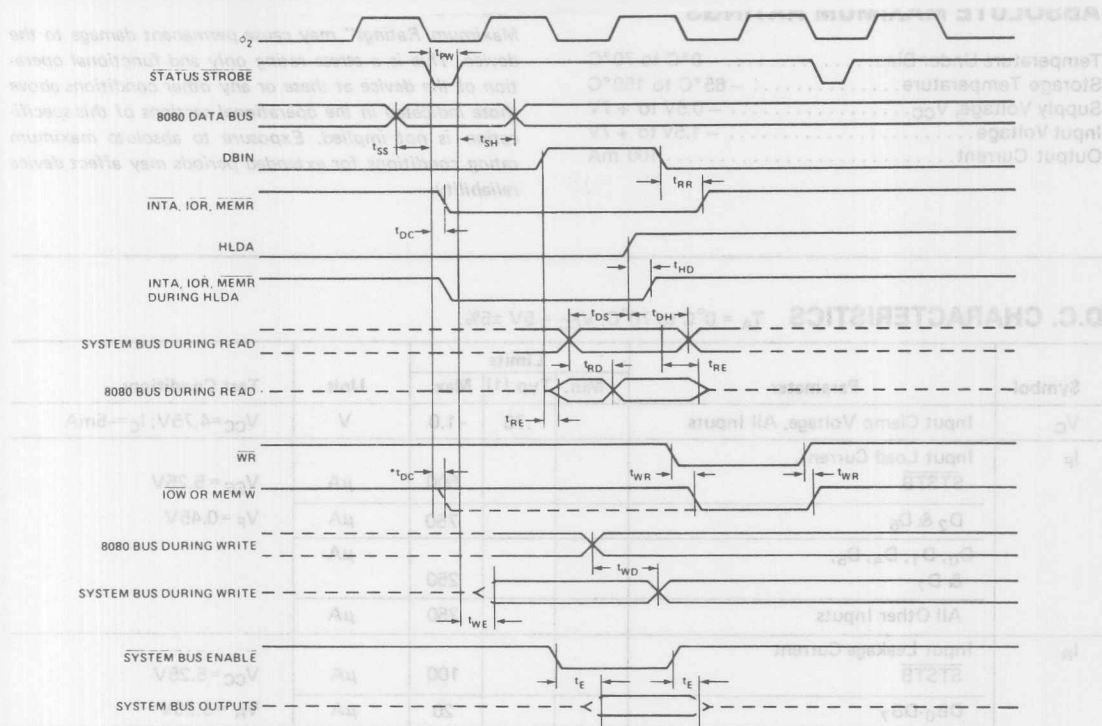
Temperature Under Bias..... -0°C to 70°C  
 Storage Temperature..... -65°C to 150°C  
 Supply Voltage,  $V_{CC}$ ..... -0.5V to +7V  
 Input Voltage..... -1.5V to +7V  
 Output Current..... 100 mA

*\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C to } 70^\circ\text{C}; V_{CC} = 5\text{V} \pm 5\%$ 

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.[1]	Max.		
$V_C$	Input Clamp Voltage, All Inputs		.75	-1.0	V	$V_{CC}=4.75\text{V}; I_C=-5\text{mA}$
$I_F$	Input Load Current, STSTB			500	$\mu\text{A}$	$V_{CC}=5.25\text{V}$
	$D_2$ & $D_6$			750	$\mu\text{A}$	$V_F=0.45\text{V}$
	$D_0, D_1, D_4, D_5,$ & $D_7$			250	$\mu\text{A}$	
	All Other Inputs			250	$\mu\text{A}$	
$I_R$	Input Leakage Current STSTB			100	$\mu\text{A}$	$V_{CC}=5.25\text{V}$
	$DB_0$ - $DB_7$			20	$\mu\text{A}$	$V_R=5.25\text{V}$
	All Other Inputs			100	$\mu\text{A}$	
$V_{TH}$	Input Threshold Voltage, All Inputs	0.8		2.0	V	$V_{CC}=5\text{V}$
$I_{CC}$	Power Supply Current		140	190	mA	$V_{CC}=5.25\text{V}$
$V_{OL}$	Output Low Voltage, $D_0$ - $D_7$			.45	V	$V_{CC}=4.75\text{V}; I_{OL}=2\text{mA}$
	All Other Outputs			.45	V	$I_{OL}=10\text{mA}$
$V_{OH}$	Output High Voltage, $D_0$ - $D_7$	3.6	3.8		V	$V_{CC}=4.75\text{V}; I_{OH}=-10\mu\text{A}$
	All Other Outputs	2.4			V	$I_{OH}=-1\text{mA}$
$I_{OS}$	Short Circuit Current, All Outputs	15		90	mA	$V_{CC}=5\text{V}$
$I_{O(off)}$	Off State Output Current, All Control Outputs			100	$\mu\text{A}$	$V_{CC}=5.25\text{V}; V_O=5.25$
				-100	$\mu\text{A}$	$V_O=.45\text{V}$
$I_{INT}$	INTA Current			5	mA	(See Figure below)

Note 1: Typical values are for  $T_A = 25^\circ\text{C}$  and nominal supply voltages.



VOLTAGE MEASUREMENT POINTS: D<sub>0</sub>-D<sub>7</sub> (when outputs) Logic "0" = 0.8V, Logic "1" = 3.0V. All other signals measured at 1.5V.

\*ADVANCED IOW/MEMW FOR 8238 ONLY.

### A.C. CHARACTERISTICS $T_A = 0^\circ\text{C}$ to $70^\circ\text{C}$ , $V_{CC} = 5V \pm 5\%$

Symbol	Parameter	Limits		Units	Condition
		Min.	Max.		
t <sub>PW</sub>	Width of Status Strobe	22		ns	
t <sub>SS</sub>	Setup Time, Status Inputs D <sub>0</sub> -D <sub>7</sub>	8		ns	
t <sub>SH</sub>	Hold Time, Status Inputs D <sub>0</sub> -D <sub>7</sub>	5		ns	
t <sub>DC</sub>	Delay from $\overline{\text{STSTB}}$ to any Control Signal	20	60	ns	C <sub>L</sub> = 100pF
t <sub>RR</sub>	Delay from DBIN to Control Outputs		30	ns	C <sub>L</sub> = 100pF
t <sub>RE</sub>	Delay from DBIN to Enable/Disable 8080 Bus		45	ns	C <sub>L</sub> = 25pF
t <sub>RD</sub>	Delay from System Bus to 8080 Bus during Read		30	ns	C <sub>L</sub> = 25pF
t <sub>WR</sub>	Delay from $\overline{\text{WR}}$ to Control Outputs	5	45	ns	C <sub>L</sub> = 100pF
t <sub>WE</sub>	Delay to Enable System Bus DB <sub>0</sub> -DB <sub>7</sub> after $\overline{\text{STSTB}}$		30	ns	C <sub>L</sub> = 100pF
t <sub>WD</sub>	Delay from 8080 Bus D <sub>0</sub> -D <sub>7</sub> to System Bus DB <sub>0</sub> -DB <sub>7</sub> during Write	5	40	ns	C <sub>L</sub> = 100pF
t <sub>E</sub>	Delay from System Bus Enable to System Bus DB <sub>0</sub> -DB <sub>7</sub>		30	ns	C <sub>L</sub> = 100pF
t <sub>HD</sub>	HLDA to Read Status Outputs		25	ns	
t <sub>DS</sub>	Setup Time, System Bus Inputs to HLDA	10		ns	
t <sub>DH</sub>	Hold Time, System Bus Inputs to HLDA	20		ns	C <sub>L</sub> = 100pF



## CAPACITANCE

This parameter is periodically sampled and not 100% tested.

Symbol	Parameter	Limits			Unit
		Min.	Typ.[1]	Max.	
C <sub>IN</sub>	Input Capacitance		8	12	pF
C <sub>OUT</sub>	Output Capacitance Control Signals		7	15	pF
I/O	I/O Capacitance (D or DB)		8	15	pF

**Test Conditions:** NS: V<sub>BIAS</sub> = 2.5V, V<sub>CC</sub> = 5.0V, T<sub>A</sub> = 25°C, f = 1MHz.

Note 2: For D<sub>0</sub>-D<sub>7</sub>: R<sub>1</sub> = 4KΩ, R<sub>2</sub> = ∞Ω,  
C<sub>L</sub> = 25pF. For all other outputs:  
R<sub>1</sub> = 500Ω, R<sub>2</sub> = 1KΩ, C<sub>L</sub> = 100pF.

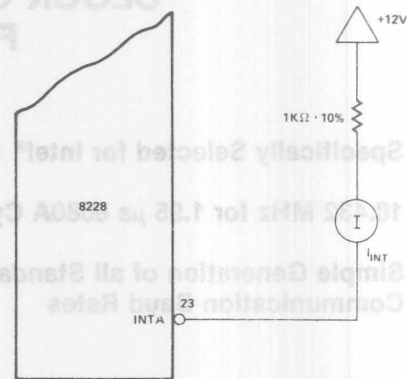
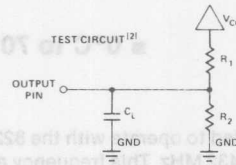


Figure 1. INTA Test Circuit (for RST 7)

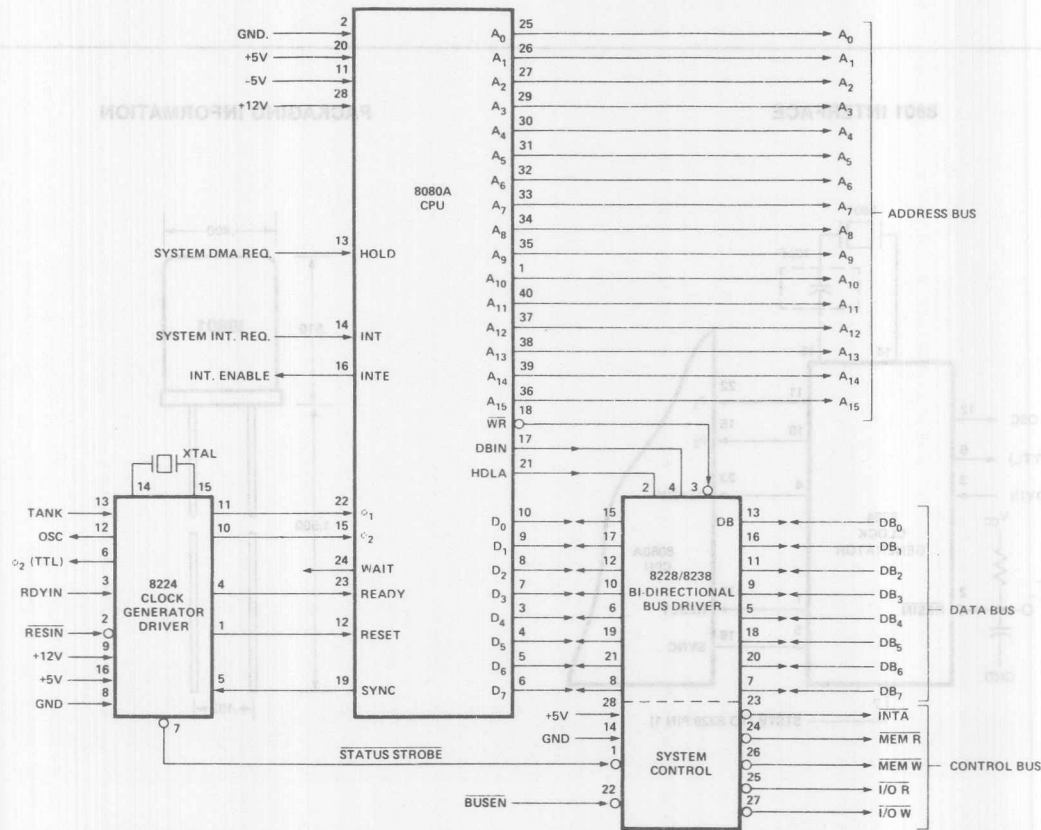


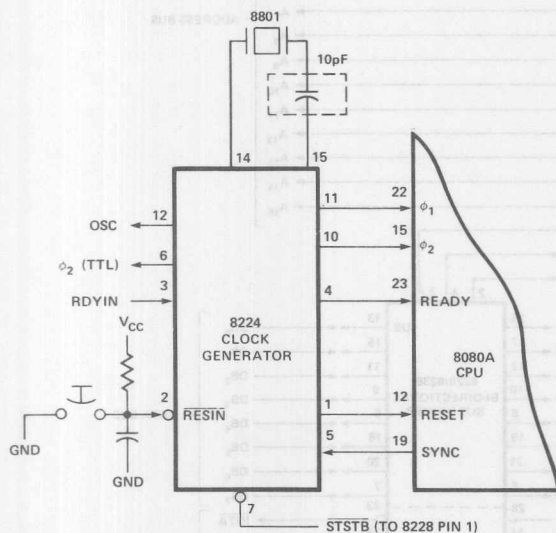
Figure 2. CPU Standard Interface

# **8801** **CLOCK GENERATOR CRYSTAL** **FOR 8224/8080A**

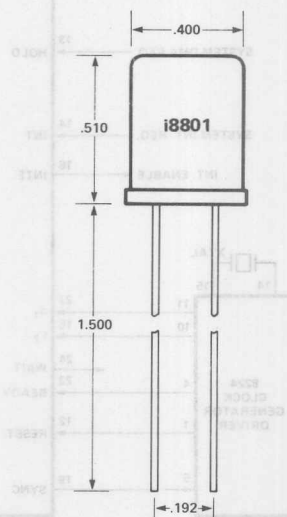
- Specifically Selected for Intel® 8224
- Frequency Deviation  $\pm 0.005\%$
- 18.432 MHz for 1.95  $\mu$ s 8080A Cycle
- Fundamental Frequency Mode
- Simple Generation of all Standard Communication Baud Rates
- 0°C to 70°C Operating Temperature

The Intel® 8801 is a quartz crystal specifically selected to operate with the 8224 clock generator and the 8080A CPU. It resonates in the fundamental frequency mode at 18.432 MHz. This frequency allows the 8080A at full speed ( $T_{CY} = 488$  ns) to have a cycle of 1.95  $\mu$ s and also simplifies the generation of all standard communication baud rates. The 8801 crystal is exactly matched to the requirements of the 8080A/8224 and provides both high performance and system flexibility for the microcomputer designer.

## **8801 INTERFACE**



## **PACKAGING INFORMATION**



## APPLICATIONS

The selection of 18.432 MHz provides the 8080A with clocks whose period is 488ns. This allows the 8080A to operate at very close to its maximum specified speed (480 ns). The 8224, when used with the 8801, outputs a signal on its OSC pin that is an approximately symmetrical square wave at a frequency of 18.432 MHz. This frequency signal can be easily divided down to generate an accurate, stable baud rate clock that can be connected directly to the transmitter or receiver clocks of the 8251 USART. This feature allows the designer to support most standard communication interfaces with a minimum of extra hardware.

The chart below (Fig. 1) shows the equivalent baud rates that are generated with the corresponding dividers.

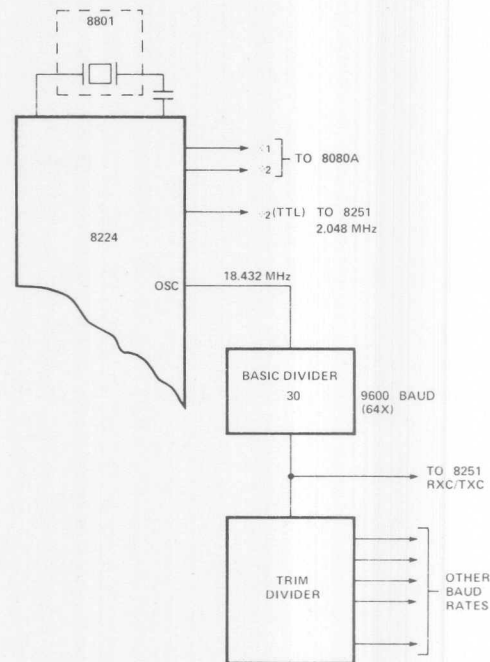


Figure 1. Block Diagram

BAUD RATE 64x	BAUD RATE 16x	FREQUENCY	BASIC DIVIDER	PLUS TRIM DIVIDER
9600		614.4 KH	÷30	—
4800	19.2K	307.2 KH	÷30	÷2
2400	9600	153.6 KH	÷30	÷4
1200	4800	76.8 KH	÷30	÷8
600	2400	38.4 KH	÷30	÷16
300	1200	19.2 KH	÷30	÷32
	600	9.6 KH	÷30	÷64
	300	4.8 KH	÷30	÷128
*109.1		6.982 KH	÷30	÷88

\*For 109.1 (64x) Baud rate divide 1200 Baud Frequency (76.8 KH) by 11.

Figure 2. Baud Rate Chart

## ELECTRICAL CHARACTERISTICS

Recommended Drive Level	5mW
Type of Resonance	Series
Equivalent Resistance	20 ohms
Maximum Shunt Capacity	7pF
Maximum Frequency Deviation	
0° — 70°C	±.005%
-55° — 125°C	±.002%



## Chapter 6

MCS-85™

MCS-80™

## Systems Support Components

Peripherals

Static RAMs

ROMs-EPROMs

ADDRESS		ENABLE		OUTPUT	
A <sub>0</sub>	A <sub>1</sub>	E <sub>0</sub>	E <sub>1</sub>	O <sub>0</sub>	O <sub>1</sub>
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0



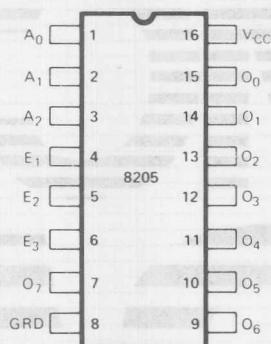
# 8205 HIGH SPEED 1 OUT OF 8 BINARY DECODER

- I/O Port or Memory Selector
- Simple Expansion — Enable Inputs
- High Speed Schottky Bipolar Technology — 18 ns Max Delay
- Directly Compatible with TTL Logic Circuits
- Low Input Load Current — 0.25 mA Max, 1/6 Standard TTL Input Load
- Minimum Line Reflection — Low Voltage Diode Input Clamp
- Outputs Sink 10 mA Min
- 16-Pin Dual In-Line Ceramic or Plastic Package

The Intel® 8205 decoder can be used for expansion of systems which utilize input ports, output ports, and memory components with active low chip select input. When the 8205 is enabled, one of its 8 outputs goes "low", thus a single row of a memory system is selected. The 3-chip enable inputs on the 8205 allow easy system expansion. For very large systems, 8205 decoders can be cascaded such that each decoder can drive 8 other decoders for arbitrary memory expansions.

The 8205 is packaged in a standard 16-pin dual in-line package, and its performance is specified over the temperature range of 0°C to +75°C, ambient. The use of Schottky barrier diode clamped transistors to obtain fast switching speeds results in higher performance than equivalent devices made with a gold diffusion process.

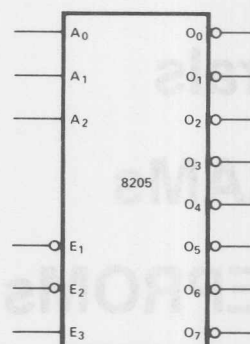
PIN CONFIGURATION



PIN NAMES

A <sub>0</sub> , A <sub>2</sub>	ADDRESS INPUTS
E <sub>1</sub> , E <sub>3</sub>	ENABLE INPUTS
O <sub>0</sub> , O <sub>7</sub>	DECODED OUTPUTS

LOGIC SYMBOL



ADDRESS			ENABLE			OUTPUTS							
A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	E <sub>1</sub>	E <sub>2</sub>	E <sub>3</sub>	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	L	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

## FUNCTIONAL DESCRIPTION

### Decoder

The 8205 contains a one out of eight binary decoder. It accepts a three bit binary code and by gating this input, creates an exclusive output that represents the value of the input code.

For example, if a binary code of 101 was present on the A0, A1 and A2 address input lines, and the device was enabled, an active low signal would appear on the  $\overline{O_5}$  output line. Note that all of the other output pins are sitting at a logic high, thus the decoded output is said to be exclusive. The decoder's outputs will follow the truth table shown below in the same manner for all other input variations.

### Enable Gate

When using a decoder it is often necessary to gate the outputs with timing or enabling signals so that the exclusive output of the decoded value is synchronous with the overall system.

The 8205 has a built-in function for such gating. The three enable inputs ( $\overline{E_1}$ ,  $\overline{E_2}$ ,  $\overline{E_3}$ ) are ANDed together and create a single enable signal for the decoder. The combination of both active "high" and active "low" device enable inputs provides the designer with a powerfully flexible gating function to help reduce package count in his system.

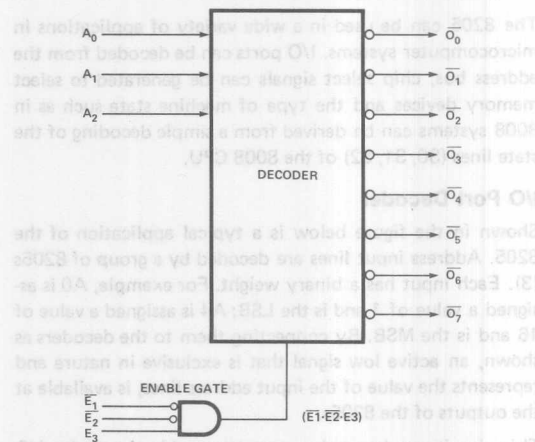


Figure 1. Enable Gate

ADDRESS			ENABLE			OUTPUTS							
A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	E <sub>1</sub>	E <sub>2</sub>	E <sub>3</sub>	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

## APPLICATIONS OF THE 8205

The 8205 can be used in a wide variety of applications in microcomputer systems. I/O ports can be decoded from the address bus, chip select signals can be generated to select memory devices and the type of machine state such as in 8008 systems can be derived from a simple decoding of the state lines (S0, S1, S2) of the 8008 CPU.

### I/O Port Decoder

Shown in the figure below is a typical application of the 8205. Address input lines are decoded by a group of 8205s (3). Each input has a binary weight. For example, A0 is assigned a value of 1 and is the LSB; A4 is assigned a value of 16 and is the MSB. By connecting them to the decoders as shown, an active low signal that is exclusive in nature and represents the value of the input address lines, is available at the outputs of the 8205s.

This circuit can be used to generate enable signals for I/O ports or any other decoder related application.

Note that no external gating is required to decode up to 24 exclusive devices and that a simple addition of an inverter or two will allow expansion to even larger decoder networks.

### Chip Select Decoder

Using a very similar circuit to the I/O port decoder, an ar-

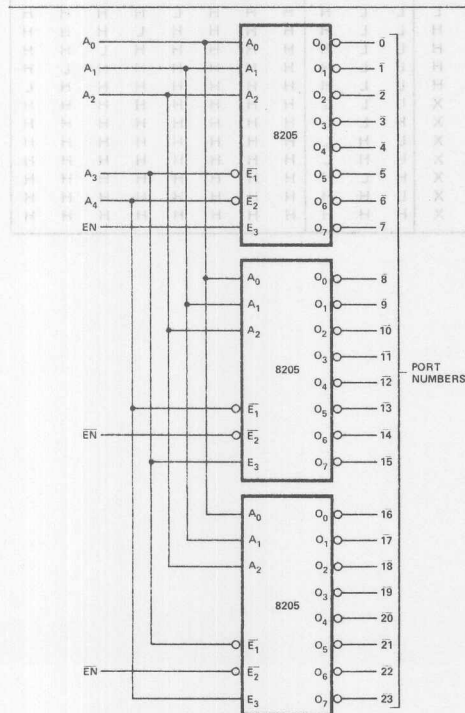


Figure 2. I/O Port Decoder

ray of 8205s can be used to create a simple interface to a 24K memory system.

The memory devices used can be either ROM or RAM and are 1K in storage capacity. 8308s and 8102s are the devices typically used for this application. This type of memory device has ten (10) address inputs and an active "low" chip select ( $\overline{CS}$ ). The lower order address bits A0-A9 which come from the microprocessor are "bussed" to all memory elements and the chip select to enable a specific device or group of devices comes from the array of 8205s. The output of the 8205 is active low so it is directly compatible with the memory components.

Basic operation is that the CPU issues an address to identify a specific memory location in which it wishes to "write" or "read" data. The most significant address bits A10-A14 are decoded by the array of 8205s and an exclusive, active low, chip select is generated that enables a specific memory device. The least significant address bits A0-A9 identify a specific location within the selected device. Thus, all addresses throughout the entire memory array are exclusive in nature and are non-redundant.

This technique can be expanded almost indefinitely to support even larger systems with the addition of a few inverters and an extra decoder (8205).

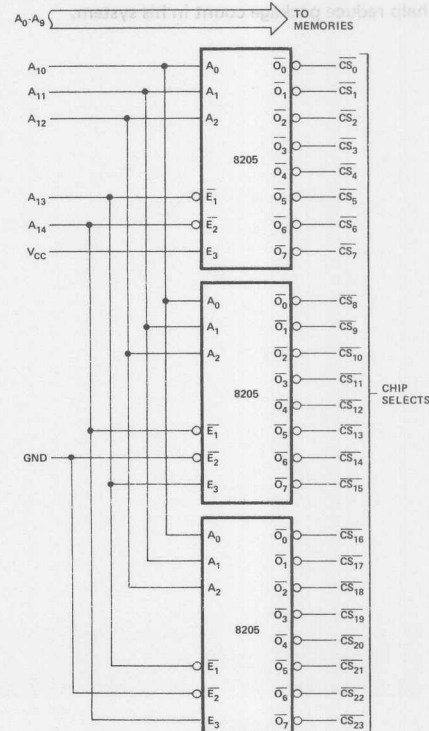


Figure 3. 32K Memory Interface

### Logic Element Example

Probably the most overlooked application of the 8205 is that of a general purpose logic element. Using the "on-chip" enabling gate, the 8205 can be configured to gate its decoded outputs with system timing signals and generate strobes that can be directly connected to latches, flip-flops and one-shots that are used throughout the system.

An excellent example of such an application is the "state decoder" in an 8008 CPU based system. The 8008 CPU issues three bits of information (S<sub>0</sub>, S<sub>1</sub>, S<sub>2</sub>) that indicate the nature of the data on the Data Bus during each machine state. Decoding of these signals is vital to generate strobes that can load the address latches, control bus discipline and general machine functions.

In the figure below a circuit is shown using the 8205 as the "state decoder" for an 8008 CPU that not only decodes the S<sub>0</sub>, S<sub>1</sub>, S<sub>2</sub> outputs but gates these signals with the clock (phase 2) and the SYNC output of the 8008 CPU. The  $\overline{T1}$

and  $\overline{T2}$  decoded strobes can connect directly to devices like 8212s for latching the address information. The other decoded strobes can be used to generate signals to control the system data bus, memory timing functions and interrupt structure. RESET is connected to the enable gate so that strobes are not generated during system reset, eliminating accidental loading.

The power of such a circuit becomes evident when a single decoded strobe is logically broken down. Consider  $\overline{T1}$  output, the boolean equation for it would be:

$$\overline{T1} = (\overline{S0} \cdot S1 \cdot S2) \cdot (\overline{SYNC} \cdot \text{Phase 2} \cdot \overline{\text{Reset}})$$

A six input NAND gate plus a few inverters would be needed to implement this function. The seven remaining outputs would need a similar circuit to duplicate their function, obviously a substantial savings in components can be achieved when using such a technique.

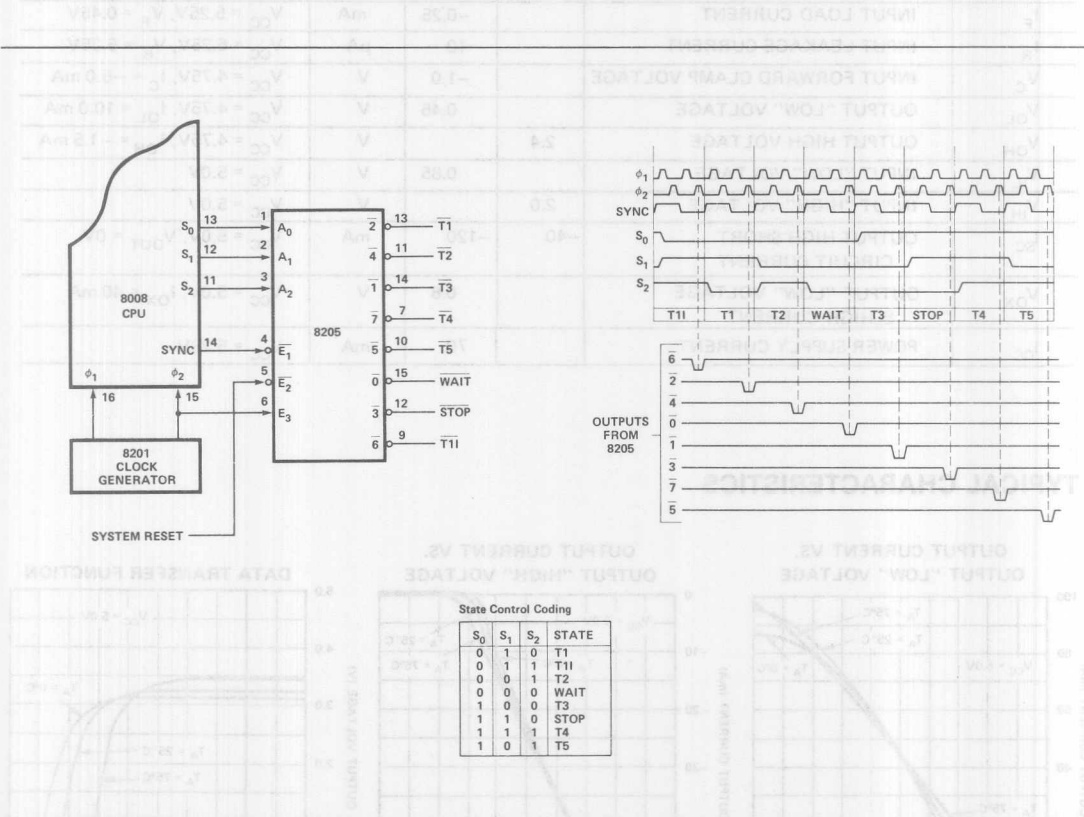


Figure 4. 8205 State Decoder Circuit

## ABSOLUTE MAXIMUM RATINGS\*

Temperature Under Bias:	Ceramic	-65°C to +125°C
	Plastic	-65°C to +75°C
Storage Temperature		-65°C to +160°C
All Output or Supply Voltages		-0.5 to +7 Volts
All Input Voltages		-1.0 to +5.5 Volts
Output Currents		125 mA

## \*COMMENT

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

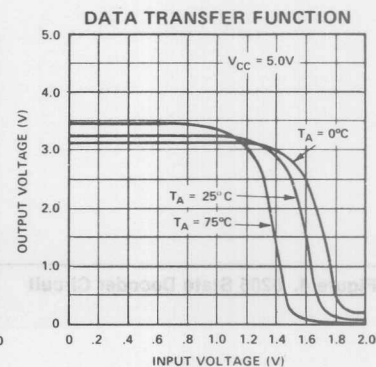
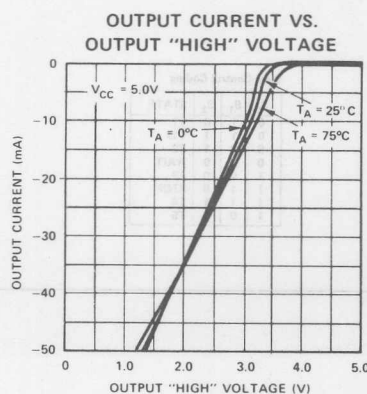
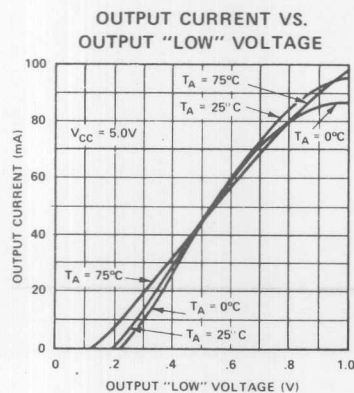
## D.C. CHARACTERISTICS

$T_A = 0^\circ\text{C}$  to  $+75^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 5\%$

## 8205

SYMBOL	PARAMETER	LIMIT		UNIT	TEST CONDITIONS
		MIN.	MAX.		
$I_F$	INPUT LOAD CURRENT		-0.25	mA	$V_{CC} = 5.25\text{V}$ , $V_F = 0.45\text{V}$
$I_R$	INPUT LEAKAGE CURRENT		10	$\mu\text{A}$	$V_{CC} = 5.25\text{V}$ , $V_R = 5.25\text{V}$
$V_C$	INPUT FORWARD CLAMP VOLTAGE		-1.0	V	$V_{CC} = 4.75\text{V}$ , $I_C = -5.0\text{mA}$
$V_{OL}$	OUTPUT "LOW" VOLTAGE		0.45	V	$V_{CC} = 4.75\text{V}$ , $I_{OL} = 10.0\text{mA}$
$V_{OH}$	OUTPUT HIGH VOLTAGE	2.4		V	$V_{CC} = 4.75\text{V}$ , $I_{OH} = -1.5\text{mA}$
$V_{IL}$	INPUT "LOW" VOLTAGE		0.85	V	$V_{CC} = 5.0\text{V}$
$V_{IH}$	INPUT "HIGH" VOLTAGE			V	$V_{CC} = 5.0\text{V}$
$I_{SC}$	OUTPUT HIGH SHORT CIRCUIT CURRENT	-40	-120	mA	$V_{CC} = 5.0\text{V}$ , $V_{OUT} = 0\text{V}$
$V_{OX}$	OUTPUT "LOW" VOLTAGE @ HIGH CURRENT		0.8	V	$V_{CC} = 5.0\text{V}$ , $I_{OX} = 40\text{mA}$
$I_{CC}$	POWER SUPPLY CURRENT		70	mA	$V_{CC} = 5.25\text{V}$

## TYPICAL CHARACTERISTICS





## SWITCHING CHARACTERISTICS

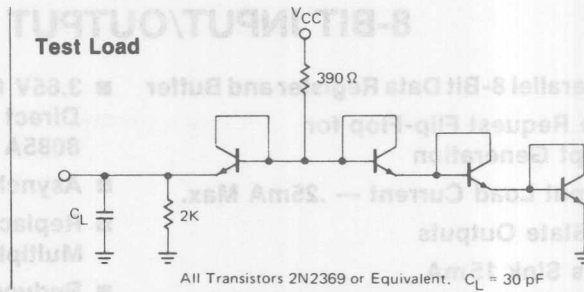
## Conditions of Test:

Input pulse amplitudes: 2.5V

Input rise and fall times: 5 nsec  
between 1V and 2V

Measurements are made at 1.5V

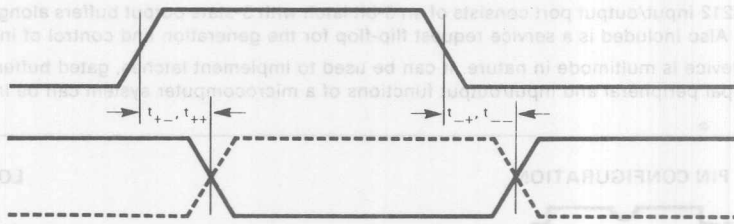
## Test Load



## Test Waveforms

ADDRESS OR ENABLE  
INPUT PULSE

OUTPUT



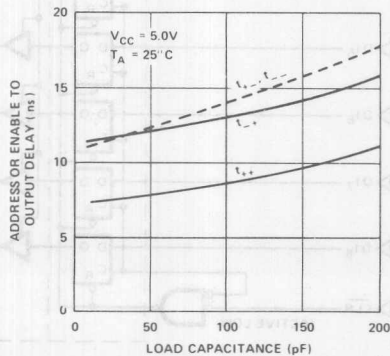
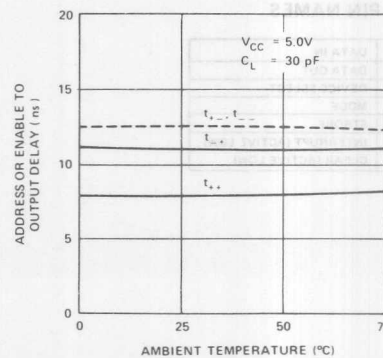
## A.C. CHARACTERISTICS

 $T_A = 0^\circ\text{C}$  to  $+75^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 5\%$  unless otherwise specified.

SYMBOL	PARAMETER	MAX. LIMIT	UNIT	TEST CONDITIONS
$t_{++}$	ADDRESS OR ENABLE TO OUTPUT DELAY	18	ns	$f = 1 \text{ MHz}$ , $V_{CC} = 0\text{V}$ $V_{BIAS} = 2.0\text{V}$ , $T_A = 25^\circ\text{C}$
$t_{-+}$		18	ns	
$t_{+-}$		18	ns	
$t_{--}$		18	ns	
$C_{IN}^{(1)}$	INPUT CAPACITANCE	P8205 4(typ.) C8205 5(typ.)	pF	

1. This parameter is periodically sampled and is not 100% tested.

## TYPICAL CHARACTERISTICS

ADDRESS OR ENABLE TO OUTPUT  
DELAY VS. LOAD CAPACITANCEADDRESS OR ENABLE TO OUTPUT  
DELAY VS. AMBIENT TEMPERATURE

# 8212

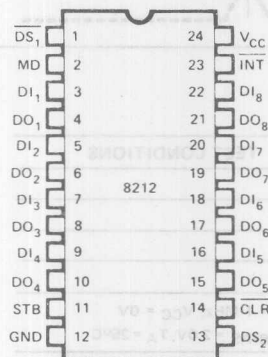
## 8-BIT INPUT/OUTPUT PORT

- Fully Parallel 8-Bit Data Register and Buffer
- Service Request Flip-Flop for Interrupt Generation
- Low Input Load Current — .25mA Max.
- Three State Outputs
- Outputs Sink 15mA
- 3.65V Output High Voltage for Direct Interface to 8008, 8080A, or 8085A CPU
- Asynchronous Register Clear
- Replaces Buffers, Latches and Multiplexers in Microcomputer Systems
- Reduces System Package Count

The 8212 input/output port consists of an 8-bit latch with 3-state output buffers along with control and device selection logic. Also included is a service request flip-flop for the generation and control of interrupts to the microprocessor.

The device is multimode in nature. It can be used to implement latches, gated buffers or multiplexers. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with this device.

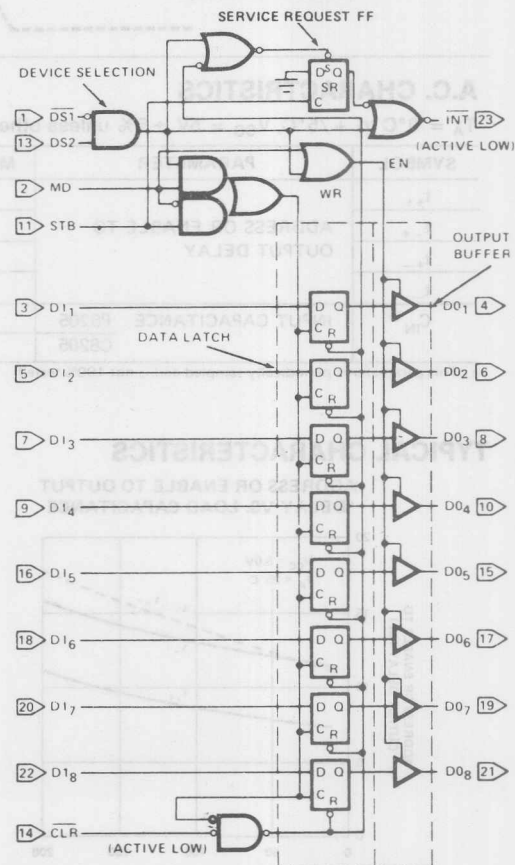
PIN CONFIGURATION



PIN NAMES

DI <sub>1</sub> DI <sub>8</sub>	DATA IN
DO <sub>1</sub> DO <sub>8</sub>	DATA OUT
DS <sub>1</sub> DS <sub>2</sub>	DEVICE SELECT
MD	MODE
STB	STROBE
INT	INTERRUPT (ACTIVE LOW)
CLR	CLEAR (ACTIVE LOW)

LOGIC DIAGRAM



## FUNCTIONAL DESCRIPTION

### Data Latch

The 8 flip-flops that make up the data latch are of a "D" type design. The output (Q) of the flip-flop will follow the data input (D) while the clock input (C) is high. Latching will occur when the clock (C) returns low.

The latched data is cleared by an asynchronous reset input ( $\overline{\text{CLR}}$ ). (Note: Clock (C) Overrides Reset ( $\overline{\text{CLR}}$ ).)

### Output Buffer

The outputs of the data latch (Q) are connected to 3-state, non-inverting output buffers. These buffers have a common control line (EN); this control line either enables the buffer to transmit the data from the outputs of the data latch (Q) or disables the buffer, forcing the output into a high impedance state. (3-state)

The high-impedance state allows the designer to connect the 8212 directly onto the microprocessor bi-directional data bus.

### Control Logic

The 8212 has control inputs  $\overline{\text{DS1}}$ ,  $\overline{\text{DS2}}$ , MD and STB. These inputs are used to control device selection, data latching, output buffer state and service request flip-flop.

### $\overline{\text{DS1}}$ , $\overline{\text{DS2}}$ (Device Select)

These 2 inputs are used for device selection. When  $\overline{\text{DS1}}$  is low and  $\overline{\text{DS2}}$  is high ( $\overline{\text{DS1}} \cdot \overline{\text{DS2}}$ ) the device is selected. In the selected state the output buffer is enabled and the service request flip-flop (SR) is asynchronously set.

### MD (Mode)

This input is used to control the state of the output buffer and to determine the source of the clock input (C) to the data latch.

When MD is high (output mode) the output buffers are enabled and the source of clock (C) to the data latch is from the device selection logic ( $\overline{\text{DS1}} \cdot \overline{\text{DS2}}$ ).

When MD is low (input mode) the output buffer state is determined by the device selection logic ( $\overline{\text{DS1}} \cdot \overline{\text{DS2}}$ ) and the source of clock (C) to the data latch is the STB (Strobe) input.

### STB (Strobe)

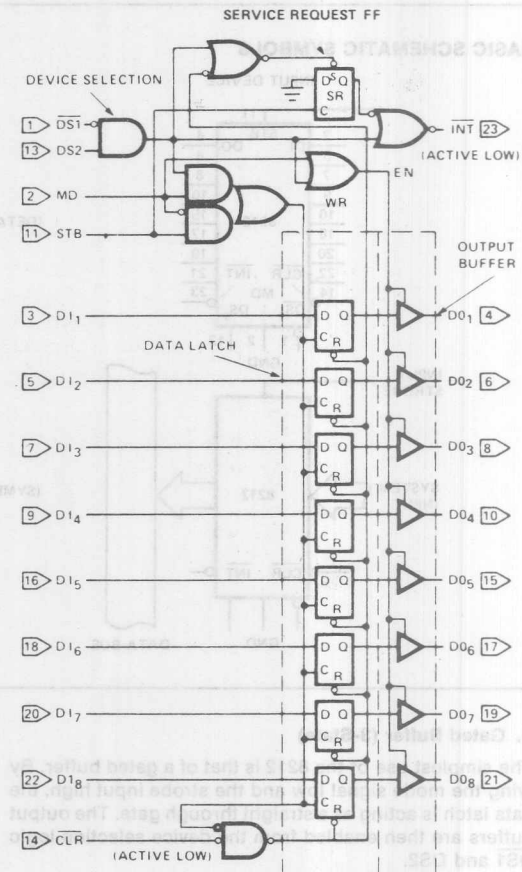
This input is used as the clock (C) to the data latch for the input mode MD = 0) and to synchronously reset the service request flip-flop (SR).

Note that the SR flip-flop is negative edge triggered.

### Service Request Flip-Flop

The (SR) flip-flop is used to generate and control interrupts in microcomputer systems. It is asynchronously set by the CLR input (active low). When the (SR) flip-flop is set it is in the non-interrupting state.

The output of the (SR) flip-flop (Q) is connected to an inverting input of a "NOR" gate. The other input to the "NOR" gate is non-inverting and is connected to the device selection logic ( $\overline{\text{DS1}} \cdot \overline{\text{DS2}}$ ). The output of the "NOR" gate ( $\overline{\text{INT}}$ ) is active low (interrupting state) for connection to active low input priority generating circuits.



STB	MD	( $\overline{\text{DS1}} \cdot \overline{\text{DS2}}$ )	DATA OUT EQUALS	CLR	( $\overline{\text{DS1}} \cdot \overline{\text{DS2}}$ )	STB	*SR	INT
0	0	0	3 STATE	0	0	0	1	1
1	0	0	3 STATE	0	1	0	1	0
0	1	0	DATA LATCH	1	1	0	0	0
1	1	0	DATA LATCH	1	1	0	1	0
0	0	1	DATA LATCH	1	0	0	1	1
1	0	1	DATA IN	1	1	0	1	0
0	1	1	DATA IN					

\*INTERNAL SR FLIP-FLOP  
CLR - RESETS DATA LATCH  
SETS SR FLIP-FLOP  
(NO EFFECT ON OUTPUT BUFFER)

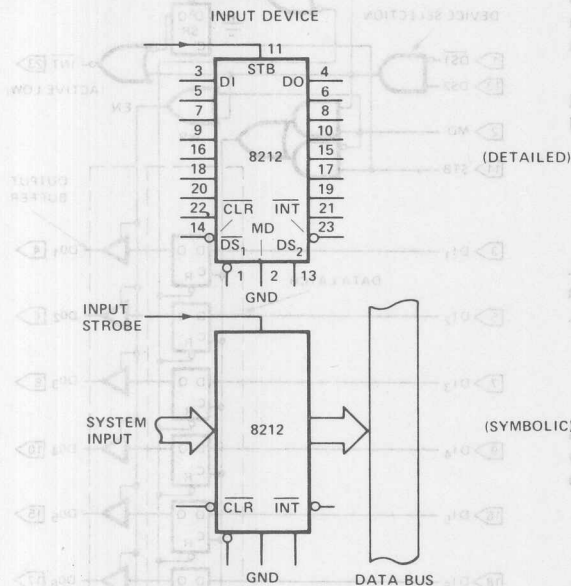
## Applications of the 8212 — For Microcomputer Systems

- I Basic Schematic Symbol
- II Gated Buffer
- III Bi-Directional Bus Driver
- IV Interrupting Input Port

### 1. Basic Schematic Symbols

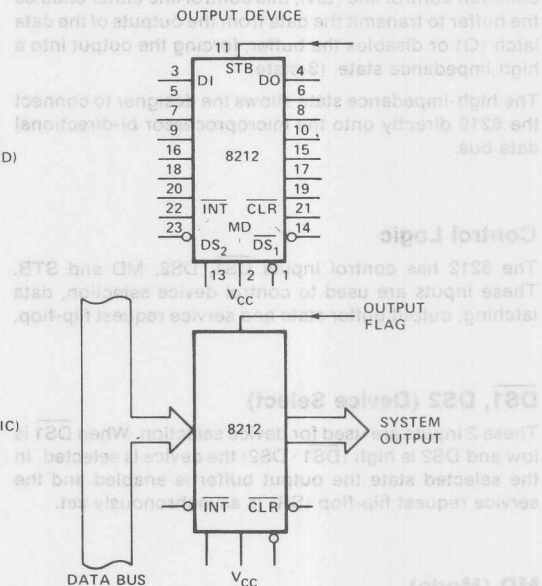
Two examples of ways to draw the 8212 on system schematics — (1) the top being the detailed view showing pin numbers, and (2) the bottom being the symbolic view

### BASIC SCHEMATIC SYMBOLS



- V Interrupt Instruction Port
- VI Output Port
- VII 8080A Status Latch
- VIII 8085A Address Latch

showing the system input or output as a system bus (bus containing 8 parallel lines). The output to the data bus is symbolic in referencing 8 parallel lines.



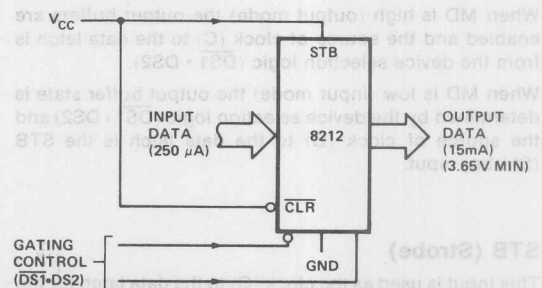
### II. Gated Buffer (3-State)

The simplest use of the 8212 is that of a gated buffer. By tying the mode signal low and the strobe input high, the data latch is acting as a straight through gate. The output buffers are then enabled from the device selection logic DS1 and DS2.

When the device selection logic is false, the outputs are 3-state.

When the device selection logic is true, the input data from the system is directly transferred to the output. The input data load is 250 micro amps. The output data can sink 15 milli amps. The minimum high output is 3.65 volts.

### GATED BUFFER



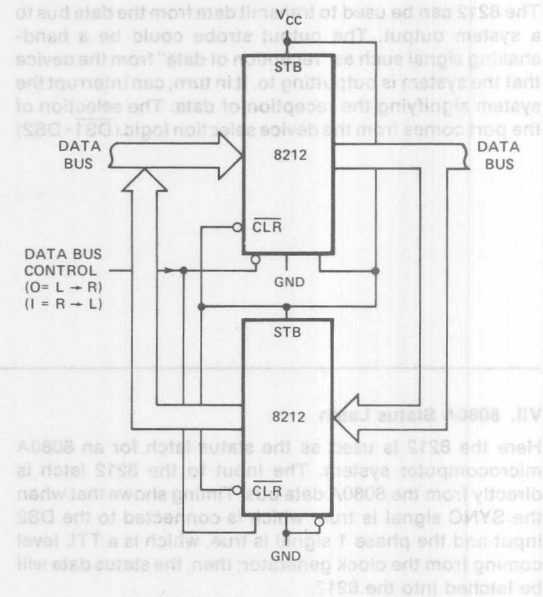
### III. Bi-Directional Bus Driver

A pair of 8212's wired (back-to-back) can be used as a symmetrical drive, bi-directional bus driver. The devices are controlled by the data bus input control which is connected to  $\overline{DS1}$  on the first 8212 and to  $\overline{DS2}$  on the second. One device is active, and acting as a straight through buffer the other is in 3-state mode. This is a very useful circuit in small system design.



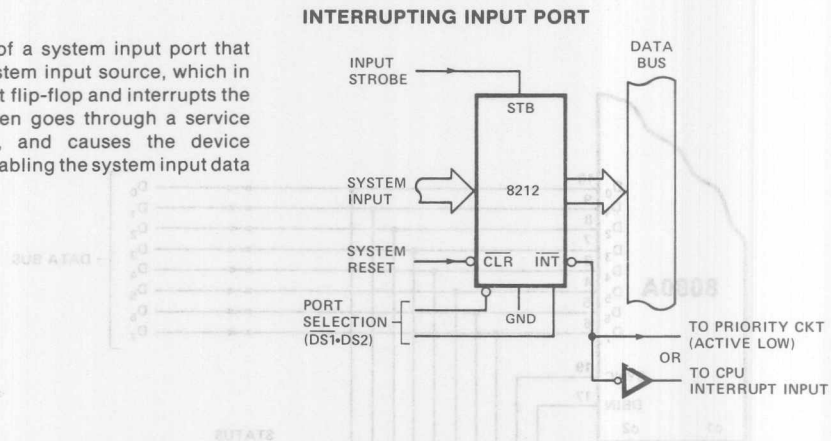
Note: The node signal is tied high in the output on the latch is active and enabled all the time. It is shown that the two areas of concern are the bi-directional data bus of the microprocessor and the control bus.

### BI-DIRECTIONAL BUS DRIVER



### IV. Interrupting Input Port

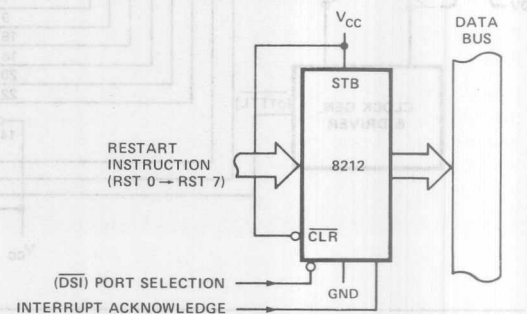
This use of an 8212 is that of a system input port that accepts a strobe from the system input source, which in turn clears the service request flip-flop and interrupts the processor. The processor then goes through a service routine, identifies the port, and causes the device selection logic to go true — enabling the system input data onto the data bus.



### V. Interrupt Instruction Port

The 8212 can be used to gate the interrupt instruction, normally RESTART instructions, onto the data bus. The device is enabled from the interrupt acknowledge signal from the microprocessor and from a port selection signal. This signal is normally tied to ground. ( $\overline{DS1}$  could be used to multiplex a variety of interrupt instruction ports onto a common bus).

### INTERRUPT INSTRUCTION PORT

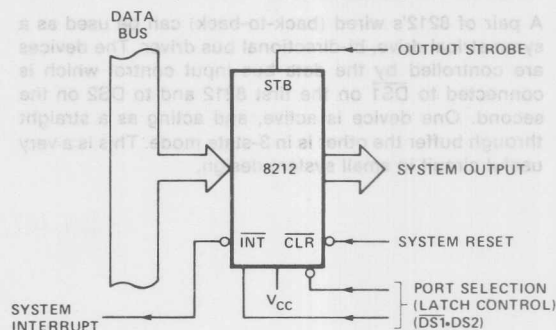




## VI. Output Port (With Hand-Shaking)

The 8212 can be used to transmit data from the data bus to a system output. The output strobe could be a handshaking signal such as "reception of data" from the device that the system is outputting to. It in turn, can interrupt the system signifying the reception of data. The selection of the port comes from the device selection logic.(DS1 • DS2)

### OUTPUT PORT (WITH HAND-SHAKING)

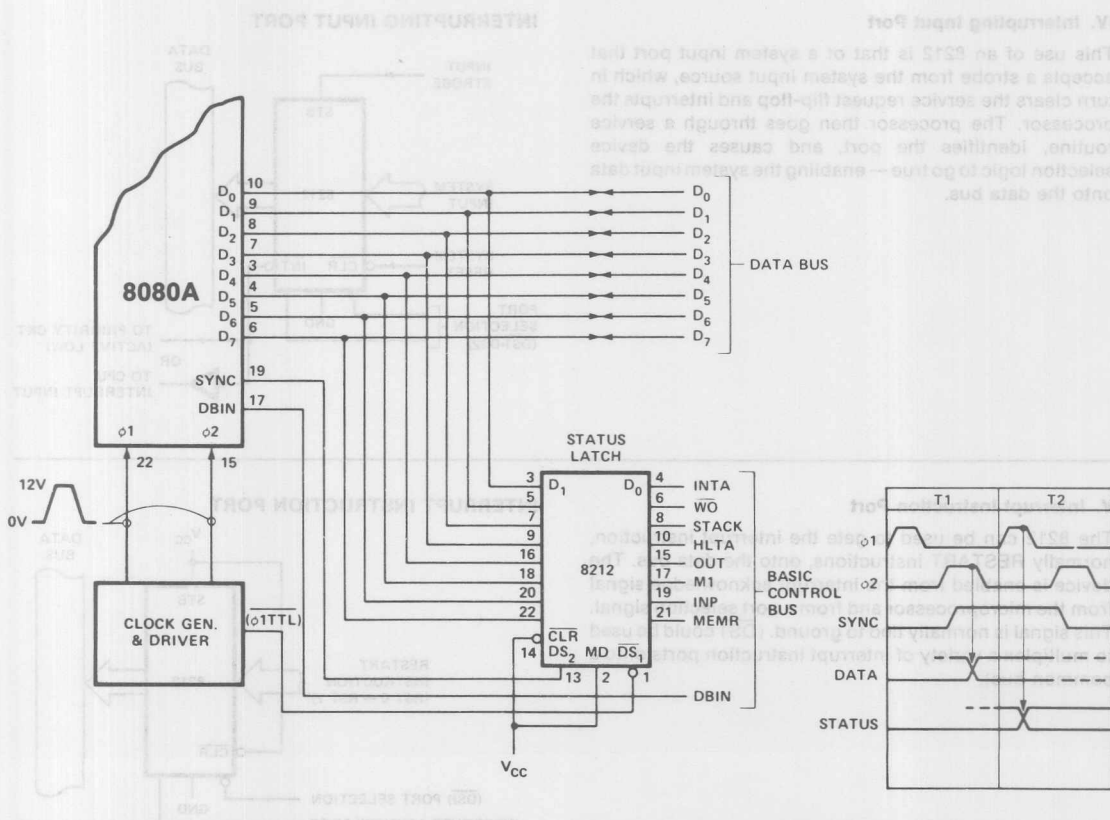


## VII. 8080A Status Latch

Here the 8212 is used as the status latch for an 8080A microcomputer system. The input to the 8212 latch is directly from the 8080A data bus. Timing shows that when the SYNC signal is true, which is connected to the DS2 input and the phase 1 signal is true, which is a TTL level coming from the clock generator; then, the status data will be latched into the 8212.

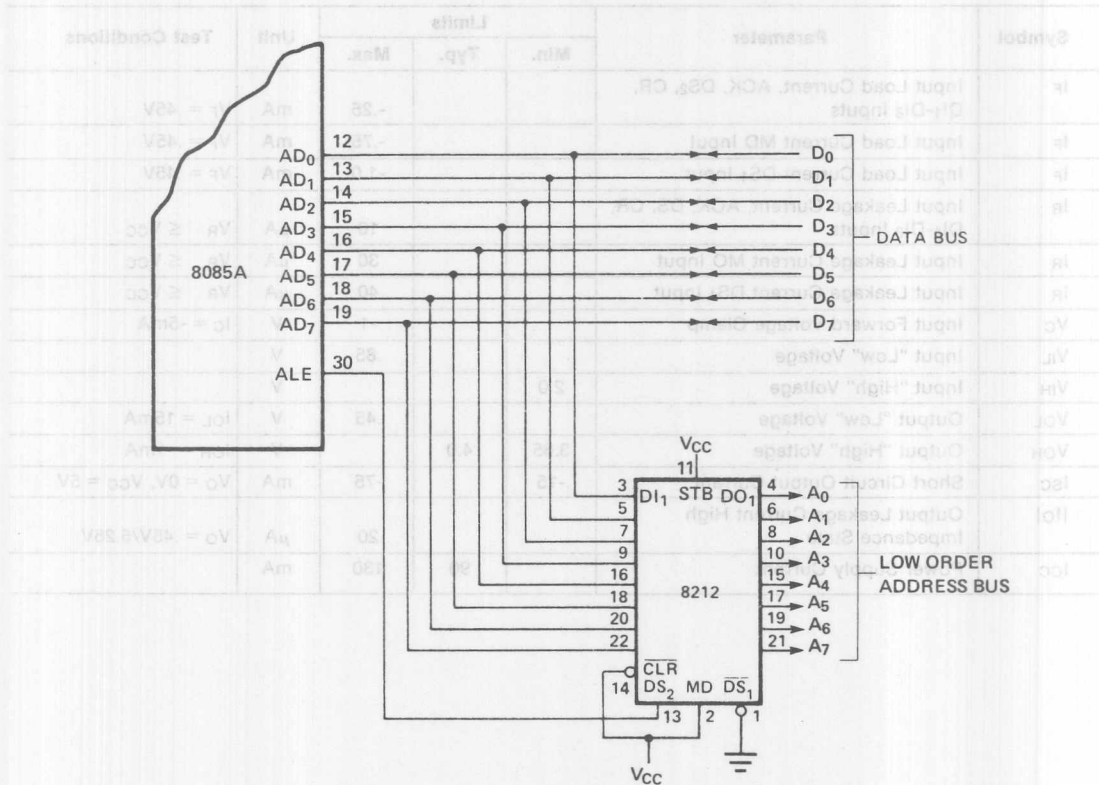
Note: The mode signal is tied high so that the output on the latch is active and enabled all the time.

It is shown that the two areas of concern are the bi-directional data bus of the microprocessor and the control bus.



### VIII. 8085A Low-Order Address Latch

The 8085A microprocessor uses a multiplexed address/data bus that contains the low order 8-bits of address information during the first part of a machine cycle. The same bus contains data at a later time in the cycle. An address latch enable (ALE) signal is provided by the 8085A to be used by the 8212 to latch the address so that it may be available through the whole machine cycle. Note: In this configuration, the MODE input is tied high, keeping the 8212's output buffers turned on at all times.

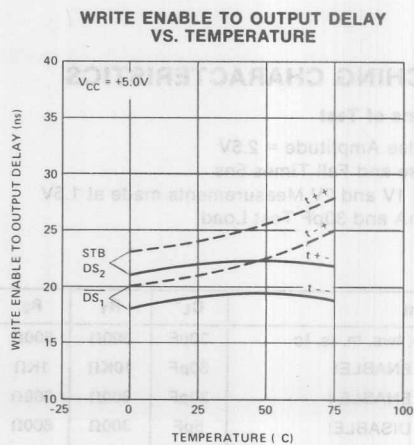
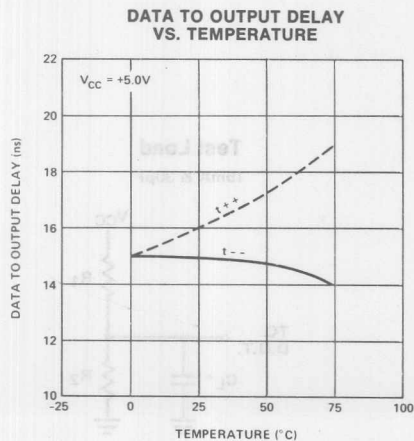
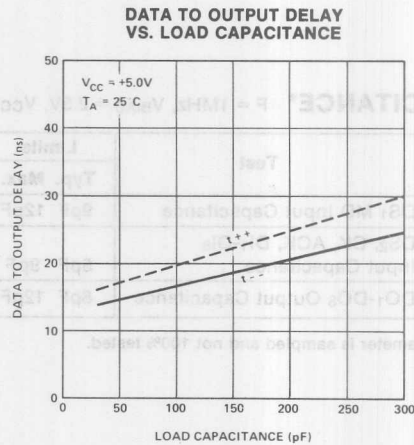
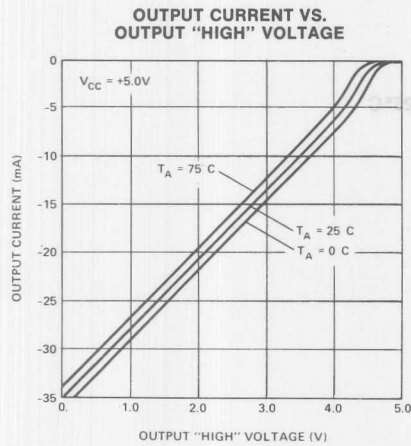
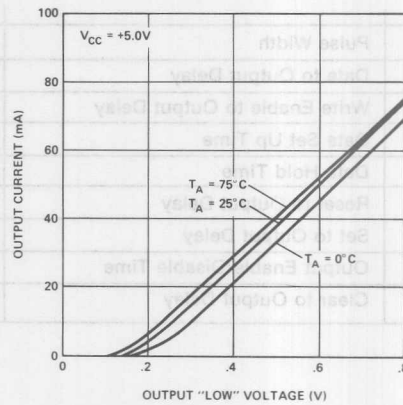
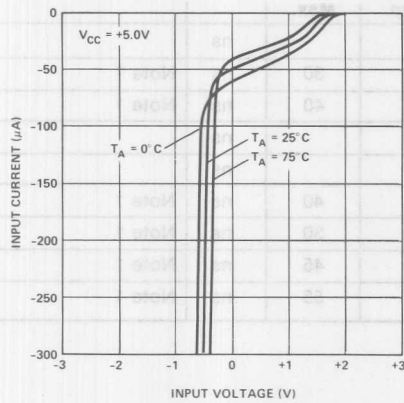


Temperature Under Bias Plastic ..... 0°C to +70°C  
Storage Temperature ..... -65°C to +160°C  
All Output or Supply Voltages ..... -0.5 to +7 Volts  
All Input Voltages ..... -1.0 to 5.5 Volts  
Output Currents ..... 100mA

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

## D.C. CHARACTERISTICS $T_A = 0^\circ\text{C}$ to $+75^\circ\text{C}$ , $V_{CC} = +5\text{V} \pm 5\%$

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
$I_F$	Input Load Current, ACK, DS <sub>2</sub> , CR, DI <sub>1</sub> -DI <sub>8</sub> Inputs			-.25	mA	$V_F = .45\text{V}$
$I_F$	Input Load Current MD Input			-.75	mA	$V_F = .45\text{V}$
$I_F$	Input Load Current DS <sub>1</sub> Input			-1.0	mA	$V_F = .45\text{V}$
$I_R$	Input Leakage Current, ACK, DS, CR, DI <sub>1</sub> -DI <sub>8</sub> Inputs			10	$\mu\text{A}$	$V_R \leq V_{CC}$
$I_R$	Input Leakage Current MO Input			30	$\mu\text{A}$	$V_R \leq V_{CC}$
$I_R$	Input Leakage Current DS <sub>1</sub> Input			40	$\mu\text{A}$	$V_R \leq V_{CC}$
$V_C$	Input Forward Voltage Clamp			-1	V	$I_C = -5\text{mA}$
$V_{IL}$	Input "Low" Voltage			.85	V	
$V_{IH}$	Input "High" Voltage	2.0			V	
$V_{OL}$	Output "Low" Voltage			.45	V	$I_{OL} = 15\text{mA}$
$V_{OH}$	Output "High" Voltage	3.65	4.0		V	$I_{OH} = -1\text{mA}$
$I_{SC}$	Short Circuit Output Current	-15		-75	mA	$V_O = 0\text{V}$ , $V_{CC} = 5\text{V}$
$ I_O $	Output Leakage Current High Impedance State			20	$\mu\text{A}$	$V_O = .45\text{V}/5.25\text{V}$
$I_{CC}$	Power Supply Current		90	130	mA	



**A.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C}$  to  $+70^\circ\text{C}$ ,  $V_{CC} = +5\text{V} \pm 5\%$ 

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
t <sub>PW</sub>	Pulse Width	30			ns	
t <sub>PD</sub>	Data to Output Delay			30	ns	Note 1
t <sub>WE</sub>	Write Enable to Output Delay			40	ns	Note 1
t <sub>SET</sub>	Data Set Up Time	15			ns	
t <sub>H</sub>	Data Hold Time	20			ns	
t <sub>R</sub>	Reset to Output Delay			40	ns	Note 1
t <sub>S</sub>	Set to Output Delay			30	ns	Note 1
t <sub>E</sub>	Output Enable/Disable Time			45	ns	Note 1
t <sub>C</sub>	Clear to Output Delay			55	ns	Note 1

**CAPACITANCE\***  $F = 1\text{MHz}$ ,  $V_{BIAS} = 2.5\text{V}$ ,  $V_{CC} = +5\text{V}$ ,  $T_A = 25^\circ\text{C}$ 

Symbol	Test	Limits	
		Typ.	Max.
C <sub>IN</sub>	DS <sub>1</sub> MD Input Capacitance	9pF	12pF
C <sub>IN</sub>	DS <sub>2</sub> , CK, ACK, DI <sub>1</sub> -DI <sub>8</sub> Input Capacitance	5pF	9pF
C <sub>OUT</sub>	DO <sub>1</sub> -DO <sub>8</sub> Output Capacitance	8pF	12pF

\*This parameter is sampled and not 100% tested.

**SWITCHING CHARACTERISTICS****Conditions of Test**

Input Pulse Amplitude = 2.5V

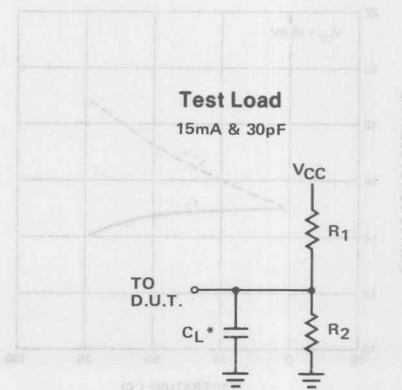
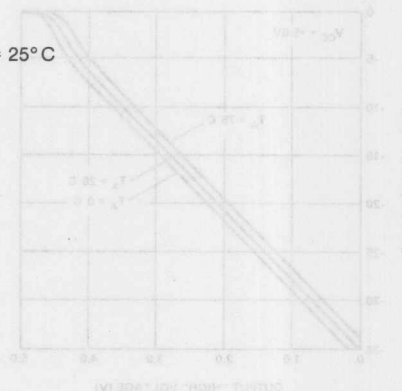
Input Rise and Fall Times 5ns

Between 1V and 2V Measurements made at 1.5V with 15mA and 30pF Test Load

Note 1:

Test	C <sub>L</sub> *	R <sub>1</sub>	R <sub>2</sub>
t <sub>PD</sub> , t <sub>WE</sub> , t <sub>R</sub> , t <sub>S</sub> , t <sub>C</sub>	30pF	300Ω	600Ω
t <sub>E</sub> , ENABLE↑	30pF	10KΩ	1KΩ
t <sub>E</sub> , ENABLE↓	30pF	300Ω	600Ω
t <sub>E</sub> , DISABLE↑	5pF	300Ω	600Ω
t <sub>E</sub> , DISABLE↓	5pF	10KΩ	1KΩ

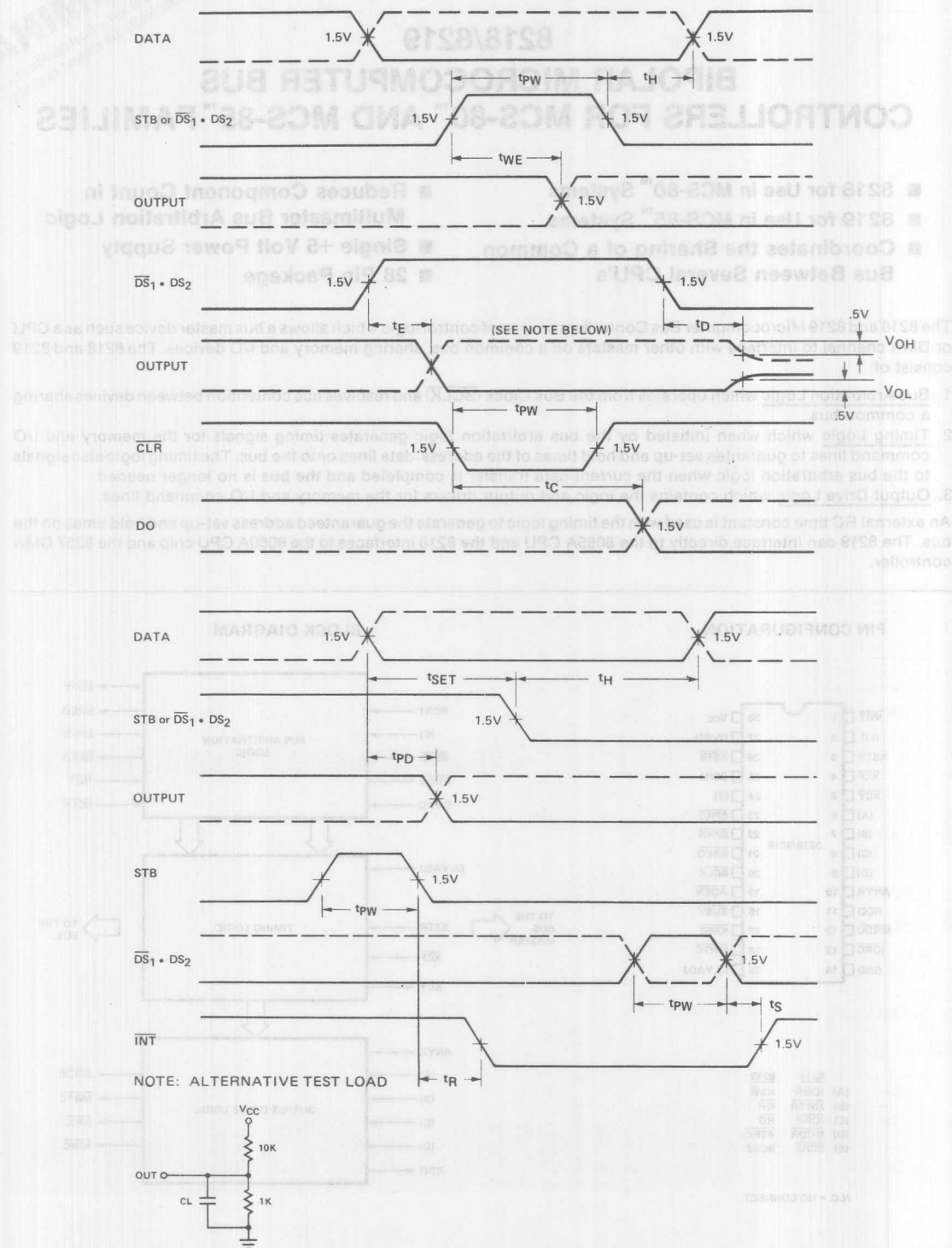
\*Includes probe and jig capacitance.



\*INCLUDING JIG & PROBE CAPACITANCE



## TIMING DIAGRAM



## 8218/8219 BIPOLAR MICROCOMPUTER BUS CONTROLLERS FOR MCS-80™ AND MCS-85™ FAMILIES

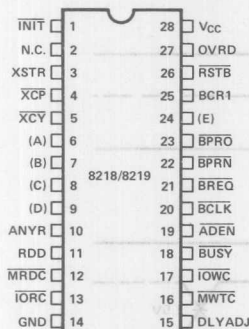
- 8218 for Use in MCS-80™ Systems
- 8219 for Use in MCS-85™ Systems
- Coordinates the Sharing of a Common Bus Between Several CPU's
- Reduces Component Count in Multimaster Bus Arbitration Logic
- Single +5 Volt Power Supply
- 28 Pin Package

The 8218 and 8219 Microcomputer Bus Controllers consist of control logic which allows a bus master device such as a CPU or DMA channel to interface with other masters on a common bus, sharing memory and I/O devices. The 8218 and 8219 consist of:

1. Bus Arbitration Logic which operates from the Bus Clock (BCLK) and resolves bus contention between devices sharing a common bus.
2. Timing Logic which when initiated by the bus arbitration logic generates timing signals for the memory and I/O command lines to guarantee set-up and hold times of the address/data lines onto the bus. The timing logic also signals to the bus arbitration logic when the current data transfer is completed and the bus is no longer needed.
3. Output Drive Logic which contains the logic and output drivers for the memory and I/O command lines.

An external RC time constant is used with the timing logic to generate the guaranteed address set-up and hold times on the bus. The 8219 can interface directly to the 8085A CPU and the 8218 interfaces to the 8080A CPU chip and the 8257 DMA controller.

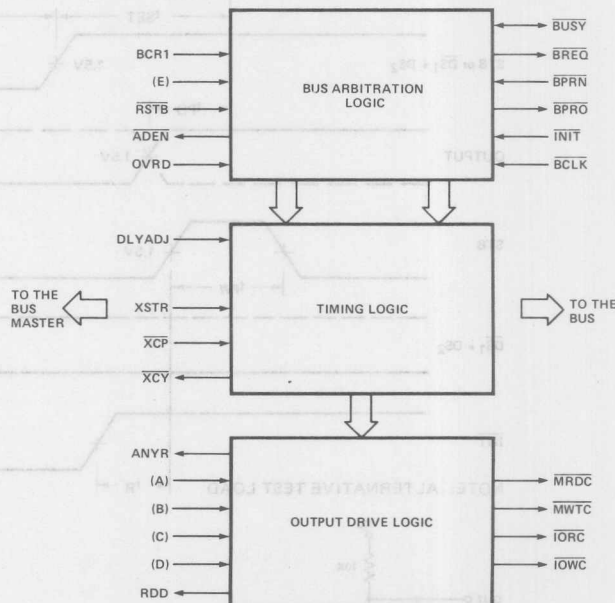
### PIN CONFIGURATION



	8218	8219
(A)	IOWR	IO/M
(B)	MWTR	WR
(C)	IORR	RD
(D)	MRDR	ASRQ
(E)	BCR2	BCR2

N.C. = NO CONNECT

### BLOCK DIAGRAM



## 8218/8219 PIN DEFINITIONS

### Signals Interfaced Directly to the System Bus

#### **BREQ (TTL Output)**

The Bus-Request is used with a central parallel priority resolution circuit. It indicates that the device needs to access the bus for one or more data transfers. It is synchronized with the Bus Clock.

#### **BUSY (Input, O.C. Output)**

Bus-Busy indicates to all master devices on the bus that the bus is in use. It inhibits any other device from getting the bus. It is synchronized with Bus Clock.

#### **BCLK (Input)**

The negative edge of Bus-Clock is used to synchronize the bus contention resolution circuit asynchronously to the CPU clock. It has 100ns min. period, 35%-65% duty cycle. It may be slowed, single stepped or stopped.

#### **BPRN (Input)**

The Bus-Priority-In indicates to a device that no device of a higher priority is requesting the bus. It is synchronous with the Bus Clock.

#### **BPRO (TTL Output)**

The Bus-Priority-Out is used with serial priority resolution circuits. Priority may be transferred to the next lower in priority as BPRN.

#### **INIT (Input)**

The Initialize resets the 8218/8219 to a known internal state.

#### **MRDC (3-State Output)**

The Memory-Read-Control indicates that the Master is requesting a read operation from the addressed location. It is asynchronous to the Bus Clock.

#### **MWTC (3-State Output)**

The Memory-Write-Control indicates that data and an address have been placed on the bus by the Master and the data is to be deposited at that location. It is asynchronous to the Bus Clock.

#### **IORC (3-State Output)**

The I/O-Read-Control indicates that the Master is requesting a read operation from the I/O device addressed. It is asynchronous to the Bus Clock.

#### **IOWC (3-State Output)**

The I/O-Write-Control indicates that Data and an I/O device address has been placed on the bus by the Master and the data is to be deposited to the I/O device.

### Signals Generated or Received by the Bus Master

#### **BCR1/BCR2 (Inputs)**

Bus-Control-Request 1 or Bus-Control-Request 2 indicate to the 8218/8219 that the Master device is making a request to control the bus. BCR2 is active low in the 8218 (BCR2). BCR2 is active high in the 8219.

#### **RSTB (Input)**

Request-Strobe latches the status of BCR1 and BCR2 into the 8218/8219. The strobe is active low in the 8218 and negative edge triggered in the 8219.

#### **ADEN (TTL Output)**

Address-and-Data-Enable indicates the Master has control of the bus. It is often used to enable Address and Data Buffers on the bus. It is synchronous with Bus Clock.

#### **RDD (TTL Output)**

Read-Data controls the direction of the bi-directional data bus drivers. It is asynchronous to the Bus Clock. A high on RDD indicates a read mode by the master.

#### **OVRD (Input)**

Override inhibits automatic deselect between transfers caused by a higher priority bus request. May be used for consecutive data transfers such as read-modify-write operations. It is asynchronous to the Bus Clock.

#### **XSTR (Input, Rising-Edge-Triggered)**

Transfer-Start-Request indicates to the 8218/8219 that a new data transfer cycle is requested to start. It is raised for each new word transfer in a multiple data word transfer. It is asynchronous to the Bus Clock.

#### **XCP (Input, Falling-Edge-Triggered)**

Transfer-Complete indicates to the 8218/8219 that the data has been received by the slave device in a write cycle or transmitted by the slave and received by master in a read cycle. It is asynchronous to the Bus Clock.

#### **XCY (TTL Output)**

Indicates that a data transfer is in progress. It is asynchronous to the Bus Clock.

#### **WR, RD, IO/M (8219 Only) (Inputs from 8085 to the 8219)**

WRITE, READ, IO/Memory are the control request inputs used by the 8085 and are internally decoded by the 8219 to produce the request signals MRDR, MWTR, IORR, IOWR. They are asynchronous to the Bus Clock.

#### **ASRQ (8219 Only) (Input from 8085 System)**

Can be used for interrupt status from the 8085. Acts like a level sensitive asynchronous bus request — no RSTB needed. It is asynchronous to the Bus Clock.

#### **MRDR, MWTR, IORR, IOWR (8218 Only) (Inputs from 8080 or 8257 to the 8218)**

Memory-Read-Request, Memory-Write-Request, I/O-Read-Request, or I/O-Write-Request indicate that address and data have been placed on the bus and the appropriate request is being made to the addressed device. Only one of these inputs should be active at any one time. They are asynchronous to the Bus Clock.

#### **ANYR (TTL Output)**

Any-Request is the logical OR of the active state of MRDR, MWTR, IORR, IOWR. It may be tied to XSTR when the rising edge of ANYR is used to initiate a transfer.

capacitor and resistor to ground to adjust the required set-up and hold time of address to control signal.

## 8218/8219 FUNCTIONAL DESCRIPTION

The 8218/8219 is a bipolar Bus Control Chip which reduces component count in the interface between a master device and the system Bus. (Master device: 8080, 8085, 8257 (DMA).)~

The 8218 and 8219 serve three major functions:

1. Resolve bus contention.
2. Guarantee set-up and hold time of address/data lines to I/O and Memory read/write control signals (adjustable by external capacitor).
3. Provide sufficient drive on all bus command lines.

### Bus Arbitration Logic

Bus Arbitration Logic begins when the Master makes a request for use of the bus on BCR1 or BCR2. The request is strobed in by RSTB. Following the next two falling edges of the bus clock (BCLK) the 8218/8219

BREQ is used for requesting the bus when priority is decided by a parallel priority resolver circuit.

BPRO is used to allow lower priority devices to gain the bus when a serial priority resolving structure is used. BPRO would go to BPRN of the next lower priority Master.

When priority is granted to the Master (a low on BPRN and a high on BUSY) the Master outputs a BUSY signal on the next falling edge of BCLK. The BUSY signal locks the master onto the bus and prohibits the enable of any other masters onto the bus.

At the same time BUSY goes active, Address and Data Enable (ADEN) goes active signifying that the Master has control of the bus. ADEN is often used to enable the bus drivers.

The Bus will be released only if the master loses priority; is not in the middle of a transfer, and Override is not active or, if the Master stops requesting the bus, is not in the middle of a data transfer, and Override is not active. ADEN then goes inactive.

Provision has been made in the 8218 to allow bus-synchronous requests. This mode is activated when BCR1, BCR2 and RSTB are all low. This action asynchronously sets the synchronization flip flop (FF2) in Figure 1a.

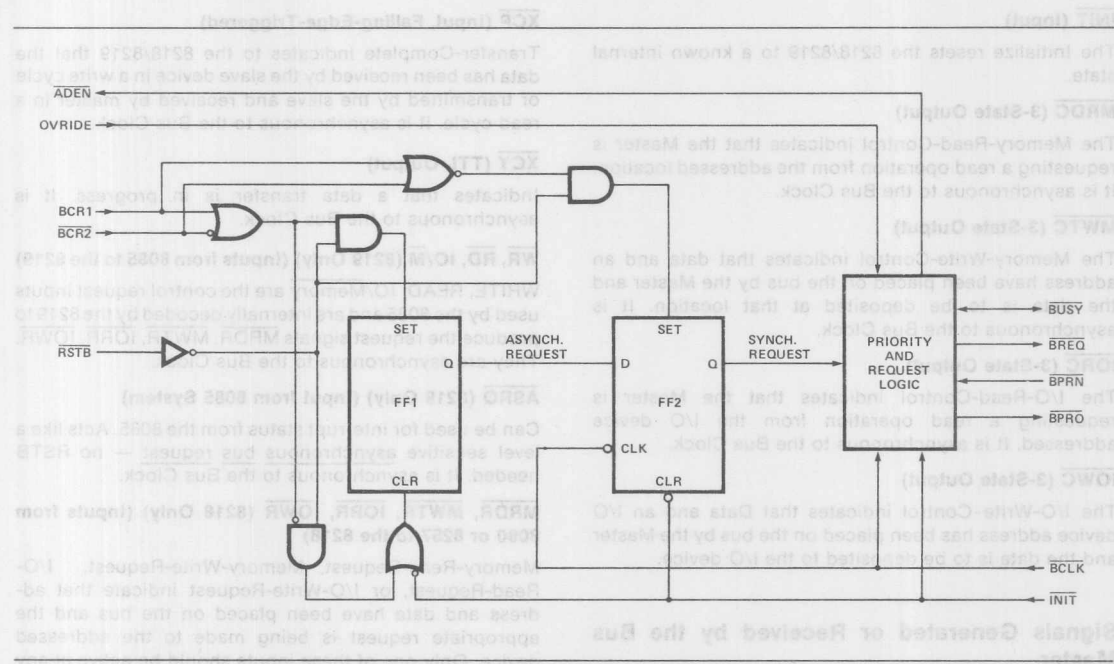


FIGURE 1a. 8218 BUS ARBITRATION LOGIC

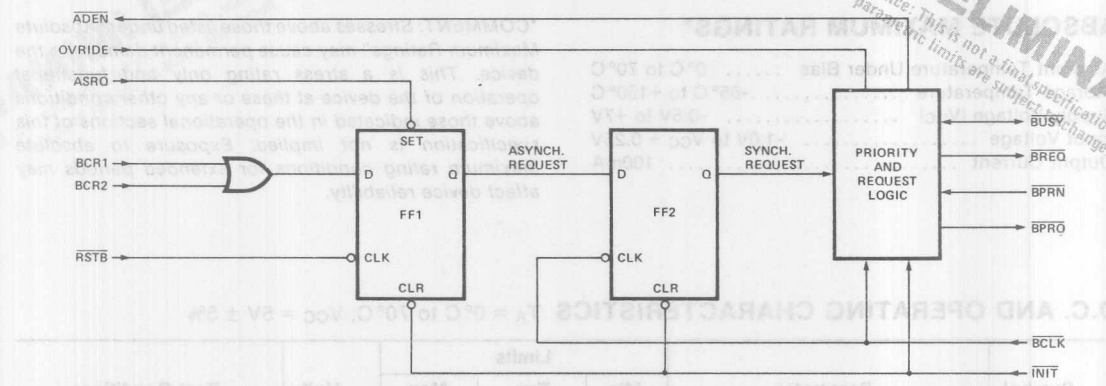


FIGURE 1b. 8219 BUS ARBITRATION LOGIC

### Timing Logic

Timing Logic activity begins with the rising edge of XSTR (Transfer Start Request) or with ADEN going active, whichever occurs second. This action causes  $\overline{XCY}$  (Transfer Cycle) to go active. 50-200ns later (depending on resistance and capacitance at DLYADJ) the appropriate Control Outputs will go active if the control input is active.

XSTR can be raised after the command goes active in the current transfer cycle so that a new transfer can be initiated immediately after the current transfer is complete.

A negative going edge on  $\overline{XCP}$  (Transfer Complete) will cause the Control Outputs (MRDC, etc.) to go inactive. 50-200ns later (depending on capacitance at DLYADJ)  $\overline{XCY}$  will go inactive indicating the transfer cycle is completed.

Additional logic within the 8218/8219 guarantees that if a transfer cycle is started ( $\overline{XCY}$  is active), but the bus is not requested (BREQ is inactive) and there is no command request input (ANYR is output low), then the transfer cycle will be cleared. This allows the bus to be released in applications where advanced bus requests are generated but the processor enters a HALT mode.

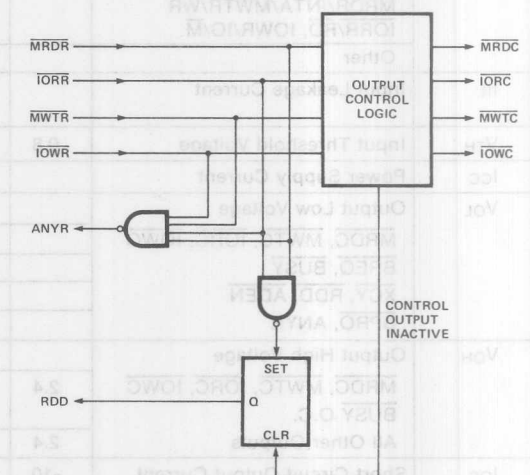


FIGURE 2a. 8218 CONTROL LOGIC

### Control Logic

The control outputs are generated in the 8219 by decoding the 8085 system control outputs (i.e., RD, WR, IO/M) or in the 8218 by directly buffering the control inputs to the control outputs for use in an 8080 or DMA system (see Figures 2a and 2b). The control outputs may be held high (inactive) by the Timing Logic. Also the control outputs are enabled when the Master gains control of the bus and disabled when control is relinquished.

The Control Logic also has two other outputs, ANYR (Any Request) and RDD (Read Data). ANYR goes high (active) if any control requests (IOWR, etc.) are active. RDD controls the direction of the Masters Bi-directional Data Bus Drivers. The Bus Driver will always be in the Write mode (RDD = Low) except from the start of a Read Control Request to 25 to 70ns after  $\overline{XCP}$  is activated.

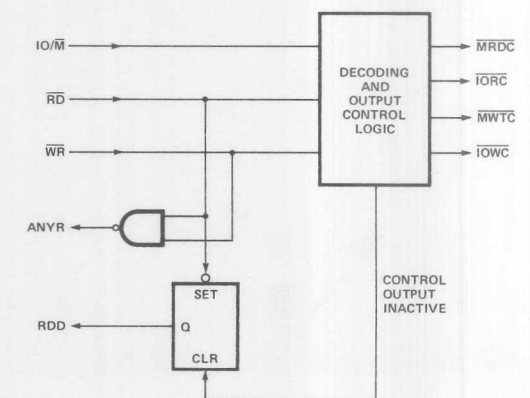


FIGURE 2b. 8219 CONTROL LOGIC



**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to +150°C
Supply Voltage (V <sub>CC</sub> )	-0.5V to +7V
Input Voltage	-1.0V to V <sub>CC</sub> + 0.25V
Output Current	100mA

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. AND OPERATING CHARACTERISTICS** T<sub>A</sub> = 0°C to 70°C; V<sub>CC</sub> = 5V ± 5%

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V <sub>C</sub>	Input Clamp Voltage			-1.0	V	V <sub>CC</sub> = 4.75V, I <sub>C</sub> = -5mA
I <sub>F</sub>	Input Load Current					V <sub>CC</sub> = 5.25V
	MRDR/INTA/MWTR/WR			-0.5	mA	V <sub>F</sub> = 0.45V
	IORR/RD, IOWR/IO/M			-0.5	mA	
I <sub>R</sub>	Input Leakage Current			100	μA	V <sub>CC</sub> = 5.25V V <sub>R</sub> = 5.25
V <sub>TH</sub>	Input Threshold Voltage	0.8		2.0	V	V <sub>CC</sub> = 5V
I <sub>CC</sub>	Power Supply Current		200	240	mA	V <sub>CC</sub> = 5.25V
V <sub>OL</sub>	Output Low Voltage					V <sub>CC</sub> = 4.75
	MRDC, MWTC, IORC, IOWC			0.45	V	I <sub>OL</sub> = 32mA
	BREQ, BUSY			0.45	V	I <sub>OL</sub> = 20mA
	XCY, RDD, ADEN			0.45	V	I <sub>OL</sub> = 16mA
	BPRO, ANYR			0.45	V	I <sub>OL</sub> = 3.2mA
V <sub>OH</sub>	Output High Voltage					V <sub>CC</sub> = 4.75V
	MRDC, MWTC, IORC, IOWC	2.4				I <sub>OH</sub> = -2mA
	BUSY O.C.	2.4				I <sub>OH</sub> = -400μA
I <sub>OS</sub>	Short Circuit Output Current	-10		-90	mA	V <sub>CC</sub> = 5.25V, V <sub>O</sub> = 0V
	Tri-State Output Current			-100	μA	V <sub>CC</sub> = 5.25V, V <sub>O</sub> = 0.45V
I <sub>O</sub> (OFF)				+100	μA	V <sub>CC</sub> = 5.25V, V <sub>O</sub> = 5.25V

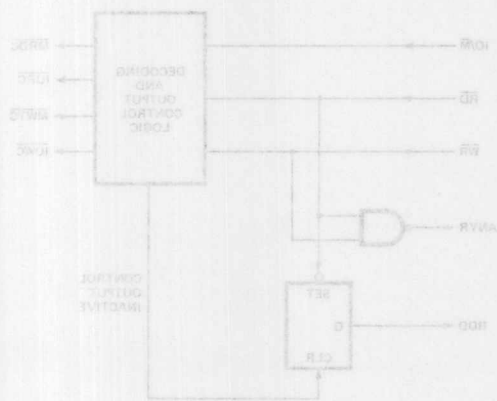


FIGURE 2B. 8218 CONTROL LOGIC

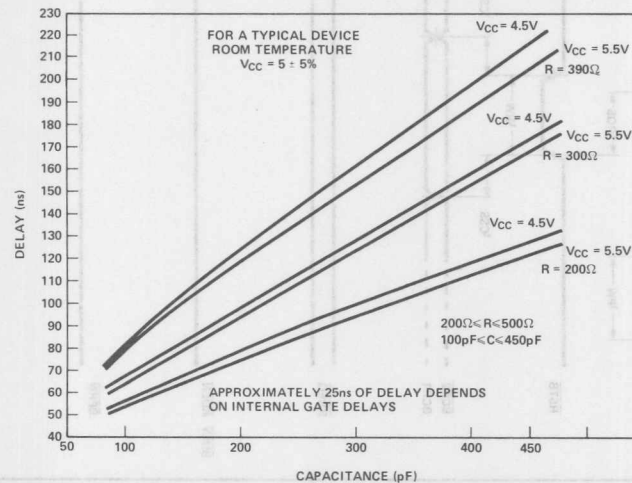
The control outputs are generated in the 8218 by decoding the 8085 system control outputs (i.e., RD, WR, IOWR) or in the 8218 by directly buffering the control inputs to the control outputs for use in an 8080 or DMA system (see Figures 2a and 2b). The control outputs may be held high (inactive) by the Timing Logic. Also the control outputs are enabled when the Master gate control of the bus and disabled when control is relinquished.

The Control Logic also has two other outputs, ANYR (Any Request) and RDD (Read Data). ANYR goes high (active) if any control requests (IOWR, etc.) are active. RDD controls the direction of the Master BI-directional Data Bus. The Bus Driver will always be in the Write mode (RDD = Low) except from the start of a Read Control Request to 25 to 70ns after XCY is activated.

**A.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5\text{V} \pm 5\%$ 

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
t <sub>BCY</sub>	Bus Clock Cycle Time	100			ns	35% to 65% Duty Cycle
t <sub>PW</sub>	Bus Clock Pulse Width	35		0.65 t <sub>BCY</sub>	ns	
t <sub>RQS</sub>	$\overline{\text{RSTB}}$ to $\overline{\text{BCLK}}$ Set-Up Time	25			ns	
t <sub>CSS</sub>	$\overline{\text{BCR}}_1$ and $\overline{\text{BCR}}_2$ to $\overline{\text{RSTB}}$ Set-Up Time	15			ns	
t <sub>CSH</sub>	$\overline{\text{BCR}}_1$ and $\overline{\text{BCR}}_2$ to $\overline{\text{RSTB}}$ Hold Time	15			ns	
t <sub>RQD</sub>	$\overline{\text{BCLK}}$ to $\overline{\text{BREQ}}$ Delay			35	ns	
t <sub>PRNS</sub>	$\overline{\text{BPRN}}$ to $\overline{\text{BCLK}}$ Set-Up Time	23			ns	
t <sub>BNO</sub>	$\overline{\text{BRPN}}$ to $\overline{\text{BPRO}}$ Delay			30	ns	
t <sub>BYD</sub>	$\overline{\text{BCLK}}$ to $\overline{\text{BUSY}}$ Delay			55	ns	
t <sub>CAD</sub>	$\overline{\text{MRDR}}$ , $\overline{\text{MWTR}}$ , $\overline{\text{IORR}}$ , $\overline{\text{IOWR}}$ to ANYR Delay			30	ns	
t <sub>SD</sub>	$\overline{\text{XSTR}}$ to $\overline{\text{XCY}}$ Delay			40	ns	
t <sub>SCD</sub>	$\overline{\text{XSTR}}$ to $\overline{\text{MRDC}}$ , $\overline{\text{MWTC}}$ , $\overline{\text{IORC}}$ , $\overline{\text{IOWC}}$ Delay	50		200	ns	Adjustable by External R/C
t <sub>SW</sub>	$\overline{\text{XSTR}}$ Pulse Width	30			ns	
t <sub>CD</sub>	$\overline{\text{XCP}}$ to $\overline{\text{MRDC}}$ , $\overline{\text{MWTC}}$ , $\overline{\text{IORC}}$ , $\overline{\text{IOWC}}$ Delay			50	ns	
t <sub>XCW</sub>	$\overline{\text{XCP}}$ Pulse Width	35			ns	
t <sub>CCD</sub>	$\overline{\text{XCP}}$ to $\overline{\text{XCY}}$ Delay	50		200	ns	Adjustable by External R/C
t <sub>CMD</sub>	$\overline{\text{MRDR}}$ , $\overline{\text{MWTR}}$ , $\overline{\text{IORR}}$ , $\overline{\text{IOWR}}$ to $\overline{\text{MRDC}}$ , $\overline{\text{MWTC}}$ , $\overline{\text{IORC}}$ , $\overline{\text{IOWC}}$			35	ns	
t <sub>CRD</sub>	$\overline{\text{MRDR}}$ , $\overline{\text{MWTR}}$ , $\overline{\text{IORR}}$ , $\overline{\text{IOWR}}$ to $\overline{\text{RDD}}$ Delay			25	ns	
t <sub>RW</sub>	$\overline{\text{RSTB}}$ Min. Neg. Pulse Width	30			ns	
t <sub>CPD</sub>	$\overline{\text{BCLK}}$ to $\overline{\text{BPRO}}$ Delay			40	ns	
t <sub>XRD</sub>	$\overline{\text{XCP}}$ to $\overline{\text{RDD}}$ Delay	25		70	ns	

**8218/19 XSTR TO OUTPUT COMMAND DELAY**  
**ONESHOT DELAY VS. DELAY ADJUST CAPACITANCE AND RESISTANCE**



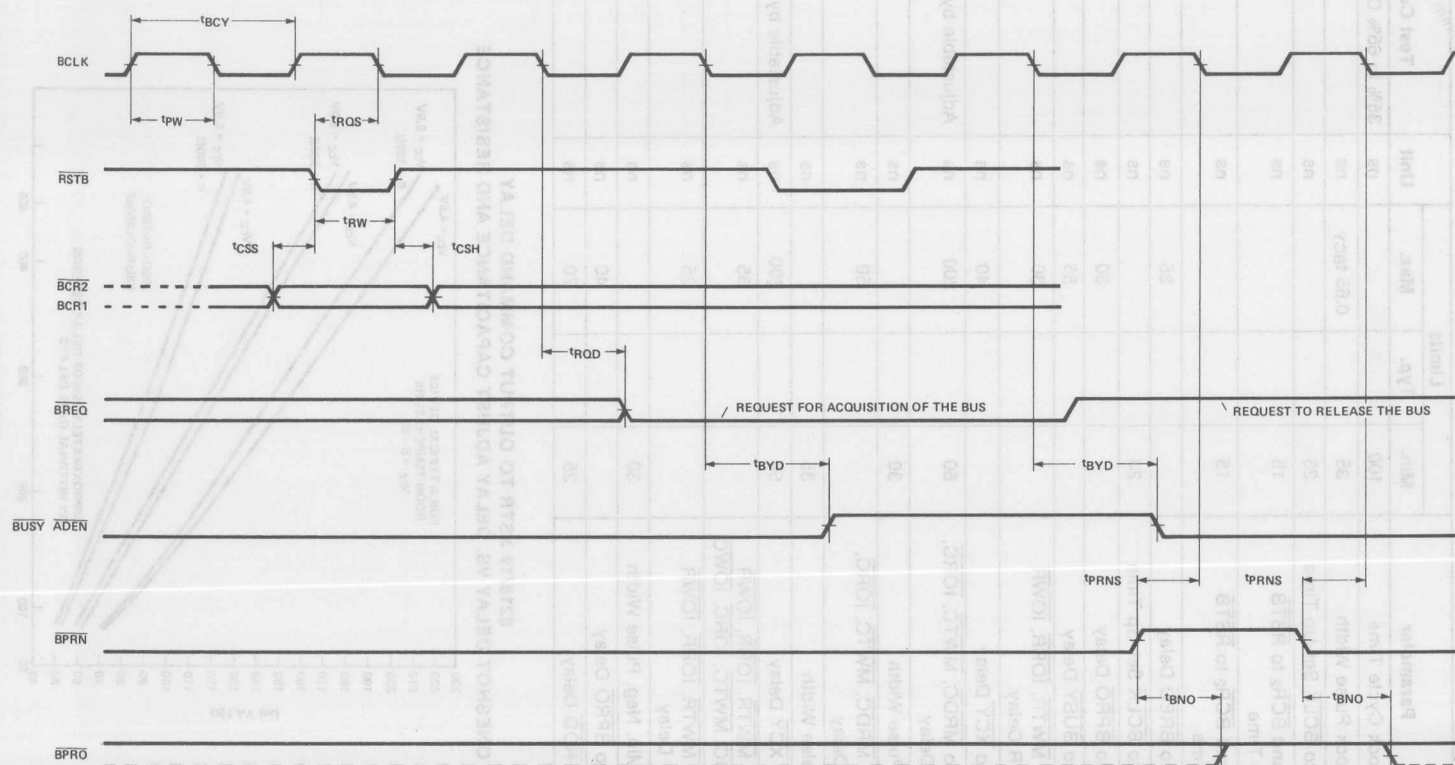


FIGURE 3a. 8218/8219 SYNCHRONOUS BUS TIMING (SYSTEM BUS PREVIOUSLY NOT IN USE).

**PRELIMINARY**  
 Notice: This is not a final specification. Some parameters and limits are subject to change.

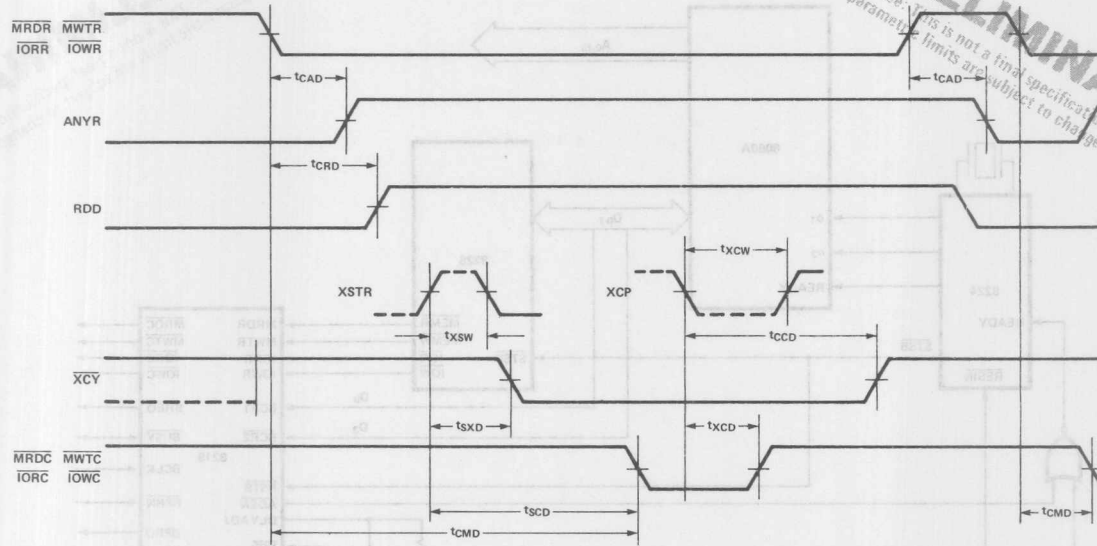


FIGURE 3b. 8218/8219 CONTROL CYCLE (SYSTEM BUS PREVIOUSLY NOT IN USE).

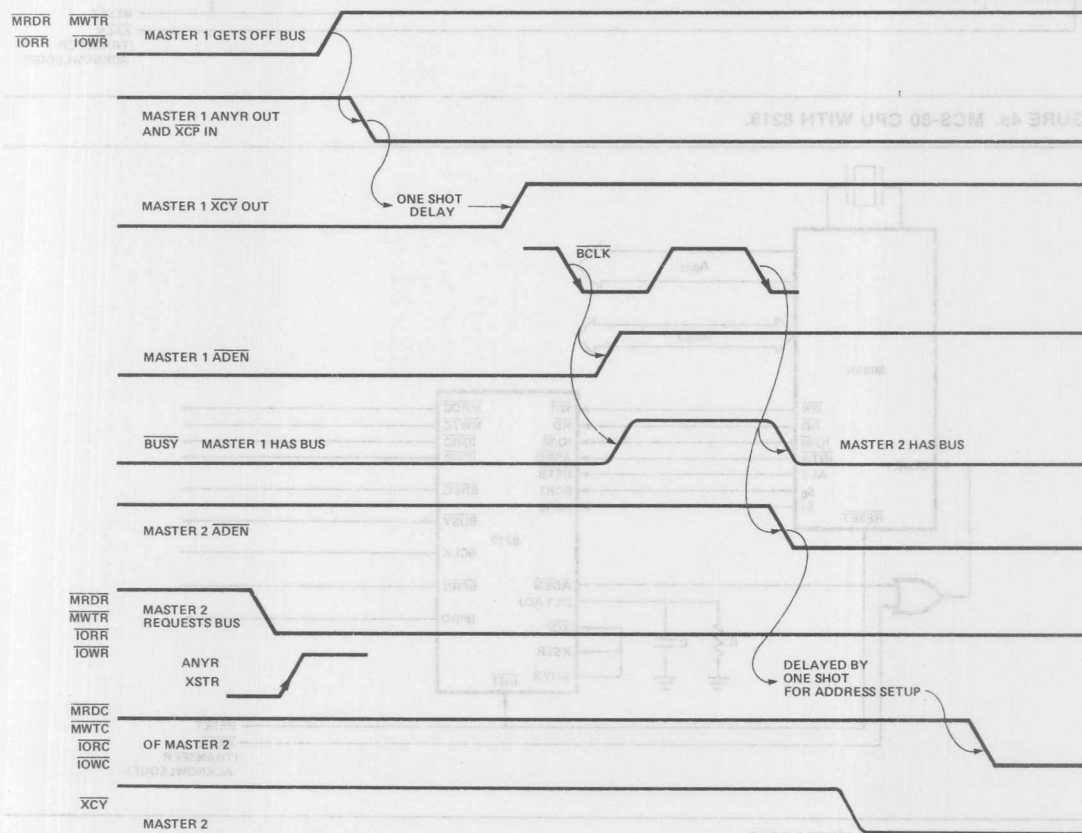


FIGURE 3c. 8218/8219 BUS CONTROL EXCHANGE (MASTER NO. 1 LEAVING BUS AND MASTER NO. 2 GETTING ON BUS).

**PRELIMINARY**  
 Notice: This is not a final specification. Some parametric limits are subject to change.

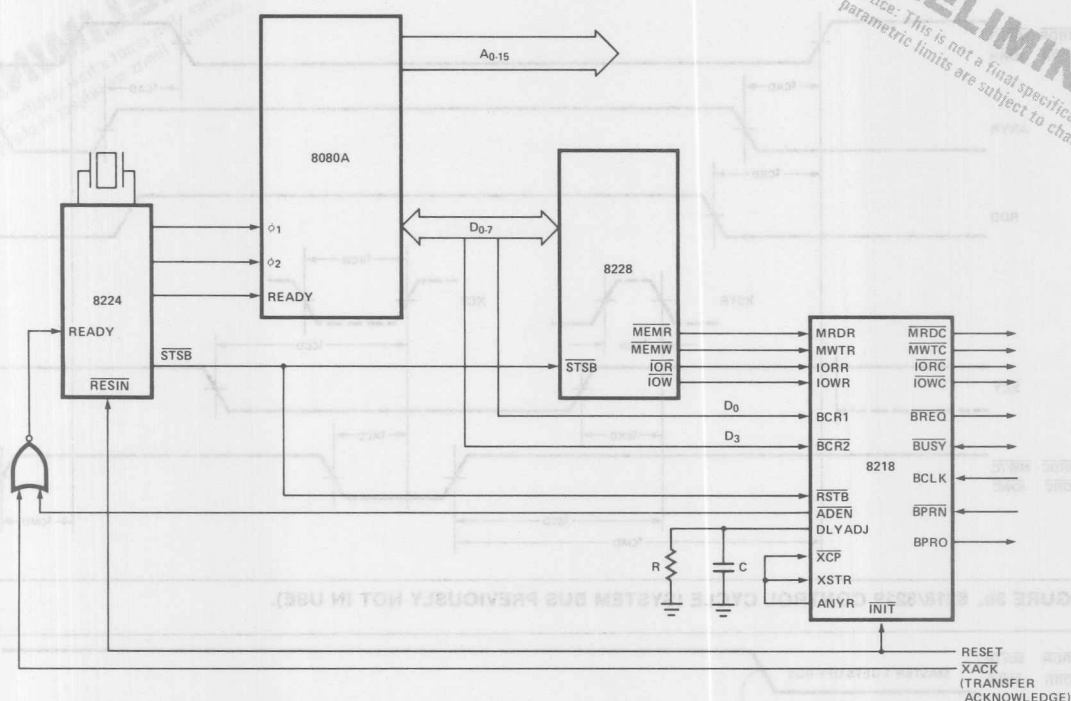


FIGURE 4a. MCS-80 CPU WITH 8218.

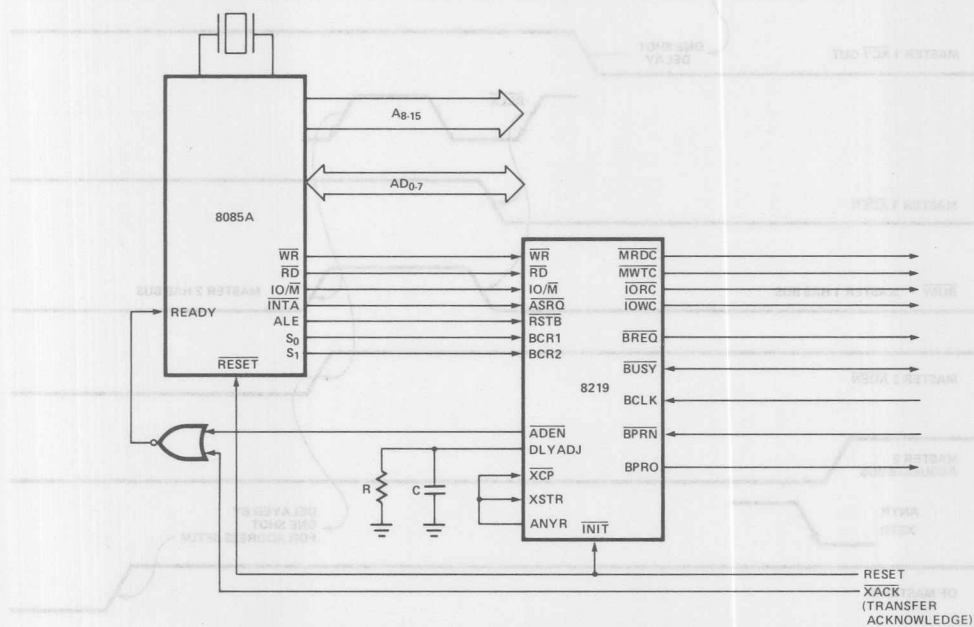
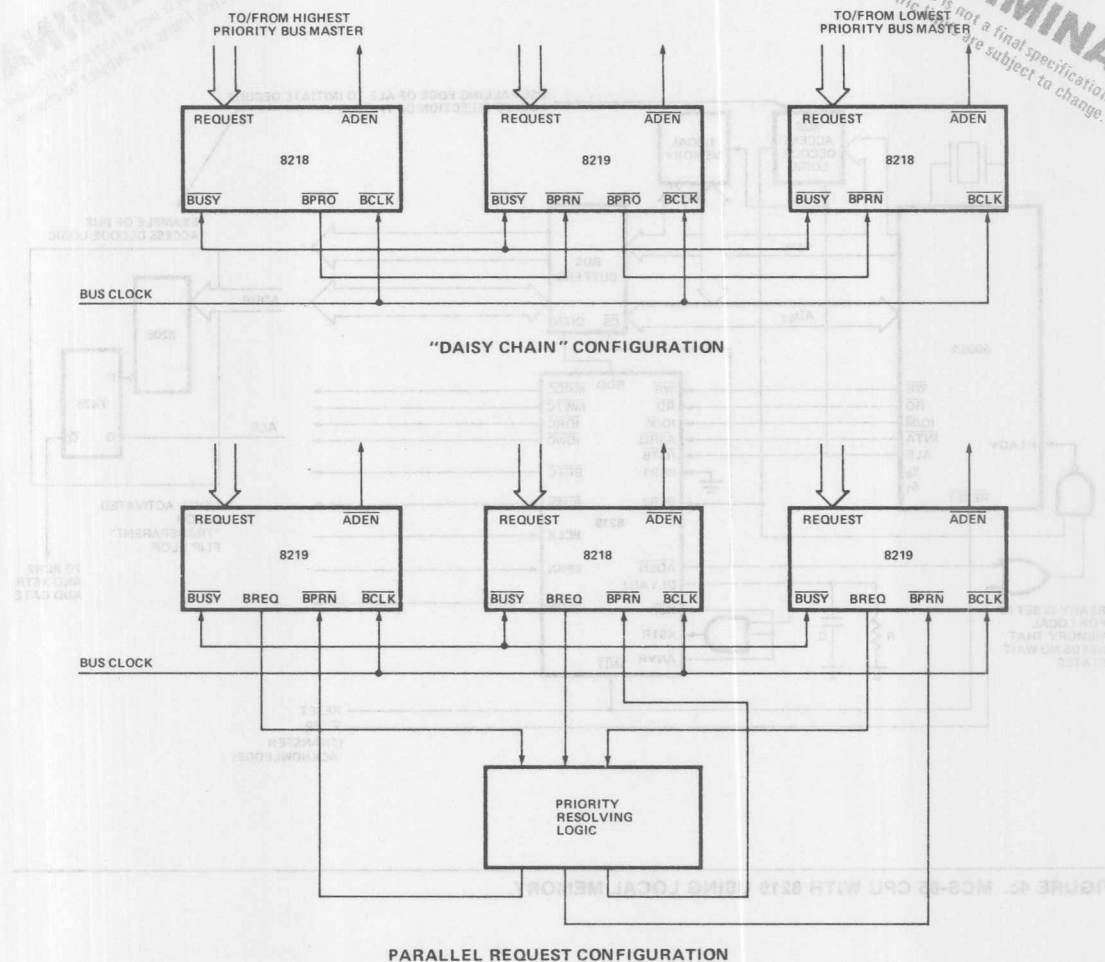


FIGURE 4b. MCS-85 CPU WITH 8219.





**PRELIMINARY**  
 Notice: This is not a final specification. Some parameters are subject to change.



**FIGURE 5. TWO METHODS OF CONNECTING MULTIPLE 8218/8219's TO RESOLVE BUS CONTENTION AMONG MULTIPLE MASTERS.**



## 8237/8237-2 HIGH PERFORMANCE PROGRAMMABLE DMA CONTROLLER

- Enable/Disable Control of Individual DMA Requests
- Four Independent DMA Channels
- Independent Autoinitialization of all Channels
- Memory-to-Memory Transfers
- Memory Block Initialization
- Address Increment or Decrement
- High Performance: Transfers up to 1.6M Bytes/Second with 5 MHz 8237-2
- Directly Expandable to any Number of Channels
- End of Process Input for Terminating Transfers
- Software DMA Requests
- Independent Polarity Control for DREQ and DACK Signals

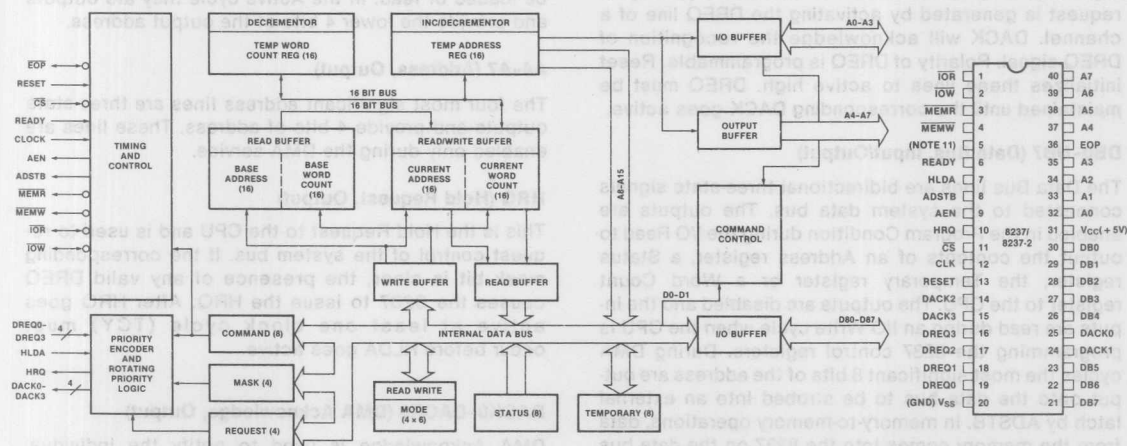
The 8237 Multimode Direct Memory Access (DMA) Controller is a peripheral interface circuit for microprocessor systems. It is designed to improve system performance by allowing external devices to directly transfer information to or from the system memory. Memory-to-memory transfer capability is also provided. The 8237 offers a wide variety of programmable control features to enhance data throughput and system optimization and to allow dynamic reconfiguration under program control.

The 8237 is designed to be used in conjunction with an external 8-bit address register such as the 8282. It contains four independent channels and may be expanded to any number of channels by cascading additional controller chips.

The three basic transfer modes allow programmability of the types of DMA service by the user. Each channel can be individually programmed to Autoinitialize to its original condition following an End of Process (EOP).

Each channel has a full 64K address and word count capability.

The 8237-2 is a 5 MHz selected version of the standard 3 MHz 8237.



BLOCK DIAGRAM

Figure 1. Pin Configuration

## PIN DEFINITIONS

**V<sub>CC</sub>:** +5 volt supply

**V<sub>SS</sub>:** Ground

### CLK (Clock, Input)

This input controls the internal operations of the 8237 and its rate of data transfers. The input may be driven at up to 3 MHz for the standard 8237 and up to 5 MHz for the 8237-2.

### $\overline{\text{CS}}$ (Chip Select, Input)

Chip Select is an active low input used to select the 8237 as an I/O device during the Idle cycle. This allows CPU communication on the data bus.

### RESET (Reset, Input)

Reset is an asynchronous active high input which clears the Command, Status, Request and Temporary registers. It also clears the first/last flip/flop and sets the Mask register. Following a Reset the device is in the Idle cycle.

### READY (Ready, Input)

Ready is an input used to extend the memory read and write pulses from the 8237 to accommodate slow memories or I/O peripheral devices.

### HLDA (Hold Acknowledge, Input)

The active high Hold Acknowledge from the CPU indicates that control of the system buses have been relinquished.

### DREQ0-DREQ3 (DMA Request, Input)

The DMA Request lines are individual asynchronous channel request inputs used by peripheral circuits to obtain DMA service. In Fixed Priority, DREQ0 has the highest priority and DREQ3 has the lowest priority. A request is generated by activating the DREQ line of a channel. DACK will acknowledge the recognition of DREQ signal. Polarity of DREQ is programmable. Reset initializes these lines to active high. DREQ must be maintained until the corresponding DACK goes active.

### DB0-DB7 (Data Bus, Input/Output)

The Data Bus lines are bidirectional three-state signals connected to the system data bus. The outputs are enabled in the Program Condition during the I/O Read to output the contents of an Address register, a Status register, the Temporary register or a Word Count register to the CPU. The outputs are disabled and the inputs are read during an I/O Write cycle when the CPU is programming the 8237 control registers. During DMA cycles the most significant 8 bits of the address are output onto the data bus to be strobed into an external latch by ADSTB. In memory-to-memory operations, data from the memory comes into the 8237 on the data bus during the read-from-memory transfer. In the write-to-memory transfer, the data bus outputs place the data into the new memory location.

### $\overline{\text{IOR}}$ (I/O Read, Input/Output)

I/O Read is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to read the control registers. In the Active cycle, it is an output control signal used by the 8237 to access data from a peripheral during a DMA Write transfer.

### $\overline{\text{IOW}}$ (I/O Write, Input/Output)

I/O Write is a bidirectional active low three-state line. In the Idle cycle, it is an input control signal used by the CPU to load information into the 8237. In the Active cycle, it is an output control signal used by the 8237 to load data to the peripheral during a DMA Read transfer.

### $\overline{\text{EOP}}$ (End of Process, Input/Output)

$\overline{\text{EOP}}$  is an active low bidirectional signal. Information concerning the completion of DMA services is available at the bidirectional  $\overline{\text{EOP}}$  pin. The 8237 allows an external signal to terminate an active DMA service. This is accomplished by pulling the  $\overline{\text{EOP}}$  input low with an external  $\overline{\text{EOP}}$  signal. The 8237 also generates a pulse when the terminal count (TC) for any channel is reached. This generates an  $\overline{\text{EOP}}$  signal which is output through the  $\overline{\text{EOP}}$  Line. The reception of  $\overline{\text{EOP}}$ , either internal or external, will cause the 8237 to terminate the service, reset the request, and, if Autoinitialize is enabled, to write the base registers to the current registers of that channel. The mask bit and TC bit in the status word will be set for the currently active channel by  $\overline{\text{EOP}}$  unless the channel is programmed for Autoinitialize. In that case, the mask bit remains clear. During memory-to-memory transfers,  $\overline{\text{EOP}}$  will be output when the TC for channel 1 occurs.  $\overline{\text{EOP}}$  should be tied high with a pull-up resistor if it is not used to prevent erroneous end of process inputs.

### A0-A3 (Address, Input/Output)

The four least significant address lines are bidirectional three-state signals. In the Idle cycle they are inputs and are used by the 8237 to address the control register to be loaded or read. In the Active cycle they are outputs and provide the lower 4 bits of the output address.

### A4-A7 (Address, Output)

The four most significant address lines are three-state outputs and provide 4 bits of address. These lines are enabled only during the DMA service.

### HRQ (Hold Request, Output)

This is the Hold Request to the CPU and is used to request control of the system bus. If the corresponding mask bit is clear, the presence of any valid DREQ causes the 8237 to issue the HRQ. After HRQ goes active at least one clock cycle (TCY) must occur before HLDA goes active.

### DACK0-DACK3 (DMA Acknowledge, Output)

DMA Acknowledge is used to notify the individual peripherals when one has been granted a DMA cycle. The sense of these lines is programmable. Reset initializes them to active low.

**AEN (Address Enable, Output)**

This output enables the 8-bit latch containing the upper 8 address bits onto the system address bus. AEN can also be used to disable other system bus drivers during DMA transfers. AEN is active HIGH.

**ADSTB (Address Strobe, Output)**

The active high Address Strobe is used to strobe the upper address byte into an external latch.

**MEMR (Memory Read, Output)**

The Memory Read signal is an active low three-state output used to access data from the selected memory location during a DMA Read or a memory-to-memory transfer.

**MEMW (Memory Write, Output)**

The Memory Write signal is an active low three-state output used to write data to the selected memory location during a DMA Write or a memory-to-memory transfer.

**FUNCTIONAL DESCRIPTION**

The 8237 block diagram includes the major logic blocks and all of the internal registers. The data interconnection paths are also shown. Not shown are the various control signals between the blocks. The 8237 contains 344 bits of internal memory in the form of registers. Figure 2 lists these registers by name and shows the size of each. A detailed description of the registers and their functions can be found under Register Description.

Name	Size	Number
Base Address Registers	16 bits	4
Base Word Count Registers	16 bits	4
Current Address Registers	16 bits	4
Current Word Count Registers	16 bits	4
Temporary Address Register	16 bits	1
Temporary Word Count Register	16 bits	1
Status Register	8 bits	1
Command Register	8 bits	1
Temporary Register	8 bits	1
Mode Registers	6 bits	4
Mask Register	4 bits	1
Request Register	4 bits	1

**Figure 2. 8237 Internal Registers**

The 8237 contains three basic blocks of control logic. The Timing Control block generates internal timing and external control signals for the 8237. The Program Command Control block decodes the various commands given to the 8237 by the microprocessor prior to servicing a DMA Request. It also decodes the Mode Control word used to select the type of DMA during the servicing. The Priority Encoder block resolves priority contention between DMA channels requesting service simultaneously.

The Timing Control block derives internal timing from the clock input. In 8237 systems this input will usually be the  $\phi 2$  TTL clock from an 8224 or CLK from an 8085A. However, any appropriate system clock will suffice.

**DMA OPERATION**

The 8237 is designed to operate in two major cycles. These are called Idle and Active cycles. Each device cycle is made up of a number of states. The 8237 can assume seven separate states, each composed of one full clock period. State I (SI) is the inactive state. It is entered when the 8237 has no valid DMA requests pending. While in SI, the DMA controller is inactive but may be in the Program Condition, being programmed by the processor. State O (SO) is the first state of a DMA service. The 8237 has requested a hold but the processor has not yet returned an acknowledge. An acknowledge from the CPU will signal that transfers may begin. S1, S2, S3 and S4 are the working states of the DMA service. If more time is needed to complete a transfer than is available with normal timing, wait states (SW) can be inserted between S2 or S3 and S4 by the use of the Ready line on the 8237.

Memory-to-memory transfers require a read-from and a write-to-memory to complete each transfer. The states, which resemble the normal working states, use two digit numbers for identification. Eight states are required for a single transfer. The first four states (S11, S12, S13, S14) are used for the read-from-memory half and the last four states (S21, S22, S23, S24) for the write-to-memory half of the transfer.

**IDLE CYCLE**

When no channel is requesting service, the 8237 will enter the Idle cycle and perform "SI" states. In this cycle the 8237 will sample the DREQ lines every clock cycle to determine if any channel is requesting a DMA service. The device will also sample CS, looking for an attempt by the microprocessor to write or read the internal registers of the 8237. When CS is low and HRQ is low, the 8237 enters the Program Condition. The CPU can now establish, change or inspect the internal definition of the part by reading from or writing to the internal registers. Address lines A0-A3 are inputs to the device and select which registers will be read or written. The IOR and IOW lines are used to select and time reads or writes. Due to the number and size of the internal registers, an internal flip-flop is used to generate an additional bit of address. This bit is used to determine the upper or lower byte of the 16-bit Address and Word Count registers. The flip-flop is reset by Master Clear or Reset. A separate software command can also reset this flip-flop.

Special software commands can be executed by the 8237 in the Program Condition. These commands are decoded as sets of addresses with the CS and IOW. The commands do not make use of the data bus. Instructions include Clear First/Last Flip-flop and Master Clear.

**ACTIVE CYCLE**

When the 8237 is in the Idle cycle and a channel requests a DMA service, the device will output an HRQ to the microprocessor and enter the Active cycle. It is in this cycle that the DMA service will take place, in one of four modes:

**Single Transfer Mode** — In Single Transfer mode the device is programmed to make one transfer only. The



word count will be decremented and the address decremented or incremented following each transfer. When the word count goes to zero, a Terminal Count (TC) will cause an Autoinitialize if the channel has been programmed to do so.

DREQ must be held active until DACK becomes active in order to be recognized. If DREQ is held active throughout the single transfer, HRQ will go inactive and release the bus to the system. It will again go active and, upon receipt of a new HLDA, another single transfer will be performed. In 8080A/8085A systems this will ensure one full machine cycle execution between DMA transfers. Details of timing between the 8237 and other bus control protocols will depend upon the characteristics of the microprocessor involved.

**Block Transfer Mode** — In Block Transfer mode the device is activated by DREQ to continue making transfers during the service until a TC, caused by word count going to zero, or an external End of Process (EOP) is encountered. DREQ need only be held active until DACK becomes active. Again, an Autoinitialization will occur at the end of the service if the channel has been programmed for it.

**Demand Transfer Mode** — In Demand Transfer mode the device is programmed to continue making transfers until a TC or external  $\overline{\text{EOP}}$  is encountered or until DREQ goes inactive. Thus transfers may continue until the I/O device has exhausted its data capacity. After the I/O device has had a chance to catch up, the DMA service is re-established by means of a DREQ. During the time between services when the microprocessor is allowed to operate, the intermediate values of address and word count are stored in the 8237 Current Address and Current Word Count registers. Only an  $\overline{\text{EOP}}$  can cause an Autoinitialize at the end of the service.  $\overline{\text{EOP}}$  is generated either by TC or by an external signal.

**Cascade Mode** — This mode is used to cascade more than one 8237 together for simple system expansion. The HRQ and HLDA signals from the additional 8237 are connected to the DREQ and DACK signals of a channel of the initial 8237. This allows the DMA requests of the additional device to propagate through the priority network circuitry of the preceding device. The priority chain is preserved and the new device must wait for its turn to acknowledge requests. Since the cascade channel in the initial device is used only for prioritizing the additional device, it does not output any address or control signals of its own. These would conflict with the outputs of the active channel in the added device. The 8237 will respond to DREQ and DACK but all other outputs except HRQ will be disabled.

Figure 3 shows two additional devices cascaded into an initial device using two of the previous channels. This forms a two level DMA system. More 8237s could be added at the second level by using the remaining channels of the first level. Additional devices can also be added by cascading into the channels of the second level devices, forming a third level.

#### TRANSFER TYPES

Each of the three active transfer modes can perform three different types of transfers. These are Read, Write

and Verify. Write transfers move data from an I/O device to the memory by activating MEMW and IOR. Read transfers move data from memory to an I/O device by activating MEMR and IOW. Verify transfers are pseudo transfers. The 8237 operates as in Read or Write transfers generating addresses, and responding to EOP, etc. However, the memory and I/O control lines all remain inactive.

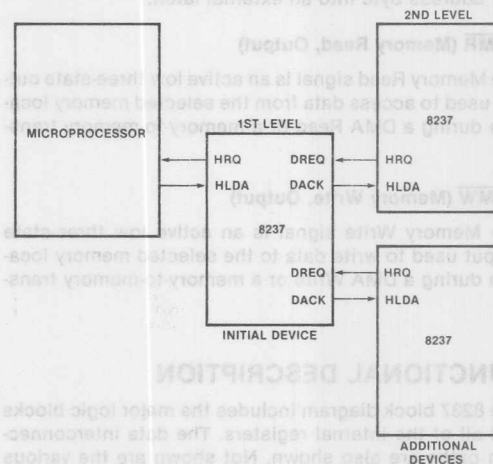


Figure 3. Cascaded 8237s

**Memory-to-Memory** — To perform block moves of data from one memory address space to another with a minimum of program effort and time, the 8237 includes a memory-to-memory transfer feature. Programming a bit in the Command register selects channels 0 and 1 to operate as memory-to-memory transfer channels. The transfer is initiated by setting the software DREQ for channel 0. The 8237 requests a DMA service in the normal manner. After HLDA is true, the device, using eight-state transfers in Block Transfer mode, reads data from the memory. The channel 0 Current Address register is the source for the address used and is decremented or incremented in the normal manner. The data byte read from the memory is stored in the 8237 internal Temporary register. Channel 1 then writes the data from the Temporary register to memory using the address in its Current Address register and incrementing or decrementing it in the normal manner. The channel 1 Current Word Count is decremented. When the word count of channel 1 goes to zero, a TC is generated causing an  $\overline{\text{EOP}}$  output, terminating the service.

Channel 0 may be programmed to retain the same address for all transfers. This allows a single word to be written to a block of memory.

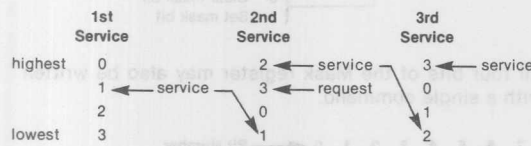
The 8237 will respond to external  $\overline{\text{EOP}}$  signals during memory-to-memory transfers. Data comparators in block search schemes may use this input to terminate the service when a match is found. The timing of memory-to-memory transfers is found in Diagram 4. Memory-to-memory operations can be detected as an active AEN with no DACK outputs.

**Autoinitialize** — By programming a bit in the Mode register, a channel may be set up as an Autoinitialize

channel. During Autoinitialize initialization, the original values of the Current Address and Current Word Count registers are automatically restored from the Base Address and Base Word Count registers of that channel following EOP. The base registers are loaded simultaneously with the current registers by the microprocessor and remain unchanged throughout the DMA service. The mask bit is not set when the channel is in Autoinitialize. Following Autoinitialize the channel is ready to perform another service without CPU intervention.

**Priority** — The 8237 has two types of priority encoding available as software selectable options. The first is Fixed Priority which fixes the channels in priority order based upon the descending value of their number. The channel with the lowest priority is 3 followed by 2, 1 and the highest priority channel, 0. After the recognition of any one channel for service, the other channels are prevented from interfering with that service until it is completed.

The second scheme is Rotating Priority. The last channel to get service becomes the lowest priority channel with the others rotating accordingly.



With Rotating Priority in a single chip DMA system, any device requesting service is guaranteed to be recognized after no more than three higher priority services have occurred. This prevents any one channel from monopolizing the system.

**Compressed Timing** — In order to achieve even greater throughput where system characteristics permit, the 8237 can compress the transfer time to two clock cycles. From Timing Diagram 3 it can be seen that state S3 is used to extend the access time of the read pulse. By removing state S3, the read pulse width is made equal to the write pulse width and a transfer consists only of state S2 to change the address and state S4 to perform the read/write. S1 states will still occur when A8-A15 need updating (see Address Generation). Timing for compressed transfers is found in Diagram 6.

**Address Generation** — In order to reduce pin count, the 8237 multiplexes the eight higher order address bits on the data lines. State S1 is used to output the higher order address bits to an external latch from which they may be placed on the address bus. The falling edge of Address Strobe (ADSTB) is used to load these bits from the data lines to the latch. Address Enable (AEN) is used to enable the bits onto the address bus through a three-state enable. The lower order address bits are output by the 8237 directly. Lines A0-A7 should be connected to the address bus. Timing Diagram 3 shows the time relationships between CLK, AEN, ADSTB, DB0-DB7 and A0-A7.

During Block and Demand Transfer mode services, which include multiple transfers, the addresses generated will be sequential. For many transfers the data held in the external address latch will remain the same. This data need only change when a carry or borrow from A7 to A8 takes place in the normal sequence of addresses. To save time and speed transfers, the 8237 executes S1 states only when updating of A8-A15 in the latch is necessary. This means for long services, S1 states may occur only once every 256 transfers, a savings of 255 clock cycles for each 256 transfers.

## REGISTER DESCRIPTION

**Current Address Register** — Each channel has a 16-bit Current Address register. This register holds the value of the address used during DMA transfers. The address is automatically incremented or decremented after each transfer and the intermediate values of the address are stored in the Current Address register during the transfer. This register is written or read by the microprocessor in successive 8-bit bytes. It may also be reinitialized by an Autoinitialize back to its original value. Autoinitialize takes place only after an EOP.

**Current Word Register** — Each channel has a 16-bit Current Word Count register. This register holds the number of transfers to be performed. The word count is decremented after each transfer. The intermediate value of the word count is stored in the register during the transfer. When the value in the register goes to zero, a TC will be generated. This register is loaded or read in successive 8-bit bytes by the microprocessor in the Program Condition. Following the end of a DMA service it may also be reinitialized by an Autoinitialization back to its original value. Autoinitialize can occur only when an EOP occurs.

**Base Address and Base Word Count Registers** — Each channel has a pair of Base Address and Base Word Count registers. These 16-bit registers store the original value of their associated current registers. During Autoinitialize these values are used to restore the current registers to their original values. The base registers are written simultaneously with their corresponding current register in 8-bit bytes in the Program Condition by the microprocessor. These registers cannot be read by the microprocessor.

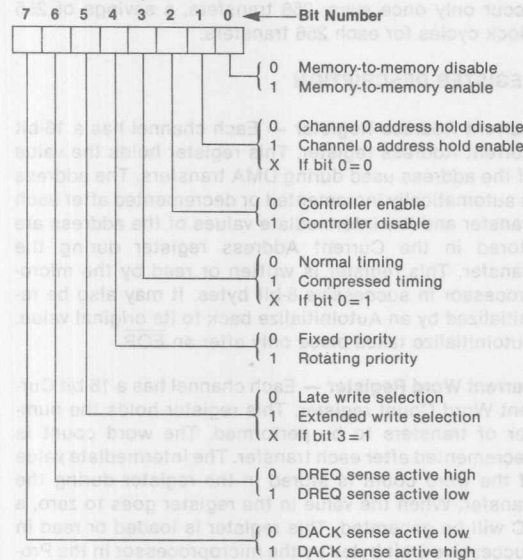
**Command Register** — This 8-bit register controls the operation of the 8237. It is programmed by the microprocessor in the Program Condition and is cleared by Reset. The following table lists the function of the command bits. See Figure 6 for address coding.

**Mode Register** — Each channel has a 6-bit Mode register associated with it. When the register is being written to by the microprocessor in the Program Condition, bits 0 and 1 determine which channel Mode register is to be written.

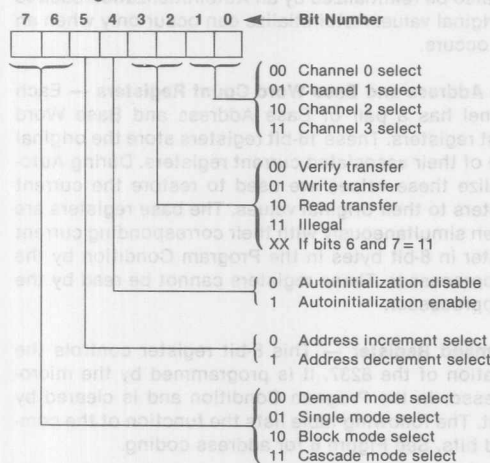
**Request Register** — The 8237 can respond to requests for DMA service which are initiated by software as well as by a DREQ. Each channel has a request bit associated with it in the 4-bit Request register. These are non-

maskable and subject to prioritization by the Priority Encoder network. Each register bit is set or reset separately under software control or is cleared upon generation of a TC or external EOP. The entire register is cleared by a Reset. To set or reset a bit, the software loads the proper form of the data word. See Figure 4 for address coding.

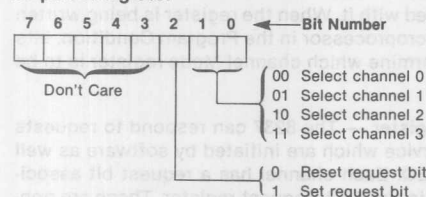
#### Command Register



#### Mode Register

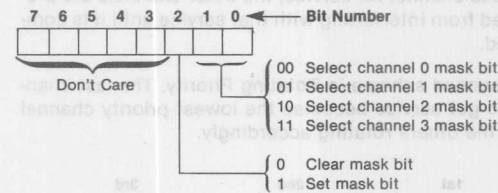


#### Request Register

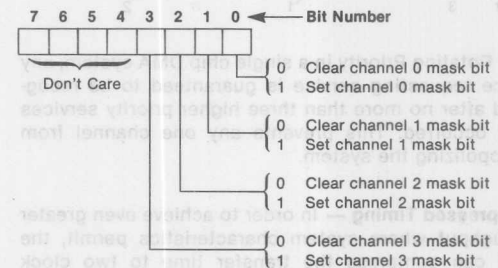


Software requests will be serviced only if the channel is in Block mode. When initiating a memory-to-memory transfer, the software request for channel 0 should be set.

**Mask Register** — Each channel has associated with it a mask bit which can be set to disable the incoming DREQ. Each mask bit is set when its associated channel produces an EOP if the channel is not programmed for Autoinitialize. Each bit of the 4-bit Mask register may also be set or cleared separately under software control. The entire register is also set by a Reset. This disables all DMA requests until a clear Mask register instruction allows them to occur. The instruction to separately set or clear the mask bits is similar in form to that used with the Request register. See Figure 4 for instruction addressing.



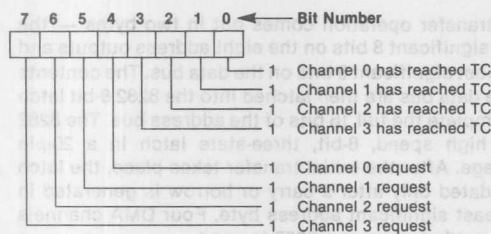
All four bits of the Mask register may also be written with a single command.



Register	Operation	Signals						
		CS	IOR	IOW	A3	A2	A1	A0
Command	Write	0	1	0	1	0	0	0
Mode	Write	0	1	0	1	0	1	1
Request	Write	0	1	0	1	0	0	1
Mask	Set/Reset	0	1	0	1	0	1	0
Mask	Write	0	1	0	1	1	1	1
Temporary	Read	0	0	1	1	1	0	1
Status	Read	0	0	1	1	0	0	0

Figure 4. Definition of Register Codes

**Status Register** — The Status register is available to be read out of the 8237 by the microprocessor. It contains information about the status of the devices at this point. This information includes which channels have reached a terminal count and which channels have pending DMA requests. Bits 0-3 are set every time a TC is reached by that channel or an external EOP is applied. These bits are cleared upon Reset and on each Status Read. Bits 4-7 are set whenever their corresponding channel is requesting service.



**Temporary Register** — The Temporary register is used to hold data during memory-to-memory transfers. Following the completion of the transfers, the last word moved can be read by the microprocessor in the Program Condition. The Temporary register always contains the last byte transferred in the previous memory-to-memory operation, unless cleared by a Reset.

**Software Commands** — These are additional special software commands which can be executed in the Program Condition. They do not depend on any specific bit pattern on the data bus. The two software commands are:

**Clear First/Last Flip-Flop:** This command is executed prior to writing or reading new address or word count information to the 8237. This initializes the flip-flop to a known state so that subsequent accesses to register contents by the microprocessor will address upper and lower bytes in the correct sequence.

**Master Clear:** This software instruction has the same effect as the hardware Reset. The Command, Status, Request, Temporary, and Internal First/Last Flip-Flop registers are cleared and the Mask register is set. The 8237 will enter the Idle cycle.

Figure 5 lists the address codes for the software commands:

Signals						Operation
A3	A2	A1	A0	IOR	IOW	
1	0	0	0	0	1	Read Status Register
1	0	0	0	1	0	Write Command Register
1	0	0	1	0	1	Illegal
1	0	0	1	1	0	Write Request Register
1	0	1	0	0	1	Illegal
1	0	1	0	1	0	Write Single Mask Register Bit
1	0	1	1	0	1	Illegal
1	0	1	1	1	0	Write Mode Register
1	1	0	0	0	1	Illegal
1	1	0	0	1	0	Clear Byte Pointer Flip/Flop
1	1	0	1	0	1	Read Temporary Register
1	1	0	1	1	0	Master Clear
1	1	1	0	0	1	Illegal
1	1	1	0	1	0	Illegal
1	1	1	1	0	1	Illegal
1	1	1	1	1	0	Write All Mask Register Bits

Figure 5. Software Command Codes

Channel	Register	Operation	Signals								Internal Flip-Flop	Data Bus DB0-DB7
			CS	IOR	IOW	A3	A2	A1	A0			
0	Base and Current Address	Write	0	1	0	0	0	0	0	0	0	A0-A7
			0	1	0	0	0	0	0	1	A8-A15	
	Current Address	Read	0	0	1	0	0	0	0	0	A0-A7	
			0	0	1	0	0	0	0	1	A8-A15	
	Base and Current Word Count	Write	0	1	0	0	0	0	1	0	W0-W7	
			0	1	0	0	0	0	1	1	W8-W15	
	Current Word Count	Read	0	0	1	0	0	0	1	0	W0-W7	
			0	0	1	0	0	0	1	1	W8-W15	
1	Base and Current Address	Write	0	1	0	0	0	1	0	0	A0-A7	
			0	1	0	0	0	1	0	1	A8-A15	
	Current Address	Read	0	0	1	0	0	1	0	0	A0-A7	
			0	0	1	0	0	1	0	1	A8-A15	
	Base and Current Word Count	Write	0	1	0	0	0	1	1	0	W0-W7	
			0	1	0	0	0	1	1	1	W8-W15	
	Current Word Count	Read	0	0	1	0	0	1	1	0	W0-W7	
			0	0	1	0	0	1	1	1	W8-W15	
2	Base and Current Address	Write	0	1	0	0	1	0	0	0	A0-A7	
			0	1	0	0	1	0	0	1	A8-A15	
	Current Address	Read	0	0	1	0	1	0	0	0	A0-A7	
			0	0	1	0	1	0	0	1	A8-A15	
	Base and Current Word Count	Write	0	1	0	0	1	0	1	0	W0-W7	
			0	1	0	0	1	0	1	1	W8-W15	
	Current Word Count	Read	0	0	1	0	1	0	1	0	W0-W7	
			0	0	1	0	1	0	1	1	W8-W15	
3	Base and Current Address	Write	0	1	0	0	1	1	0	0	A0-A7	
			0	1	0	0	1	1	0	1	A8-A15	
	Current Address	Read	0	0	1	0	1	1	0	0	A0-A7	
			0	0	1	0	1	1	0	1	A8-A15	
	Base and Current Word Count	Write	0	1	0	0	1	1	1	0	W0-W7	
			0	1	0	0	1	1	1	1	W8-W15	
	Current Word Count	Read	0	0	1	0	1	1	1	0	W0-W7	
			0	0	1	0	1	1	1	1	W8-W15	

Figure 6. Word Count and Address Register Command Codes



## APPLICATION INFORMATION

Figure 7 shows a convenient method for configuring a DMA system with the 8237 controller and an 8080A/8085A microprocessor system. The multimode DMA controller issues a HRQ to the processor whenever there is at least one valid DMA request from a peripheral device. When the processor replies with a HLDA signal, the 8237 takes control of the address bus, the data bus and the control bus. The address for the

first transfer operation comes out in two bytes — the least significant 8 bits on the eight address outputs and the most significant 8 bits on the data bus. The contents of the data bus are then latched into the 8282 8-bit latch to complete the full 16 bits of the address bus. The 8282 is a high speed, 8-bit, three-state latch in a 20-pin package. After the initial transfer takes place, the latch is updated only after a carry or borrow is generated in the least significant address byte. Four DMA channels are provided when one 8237 is used.

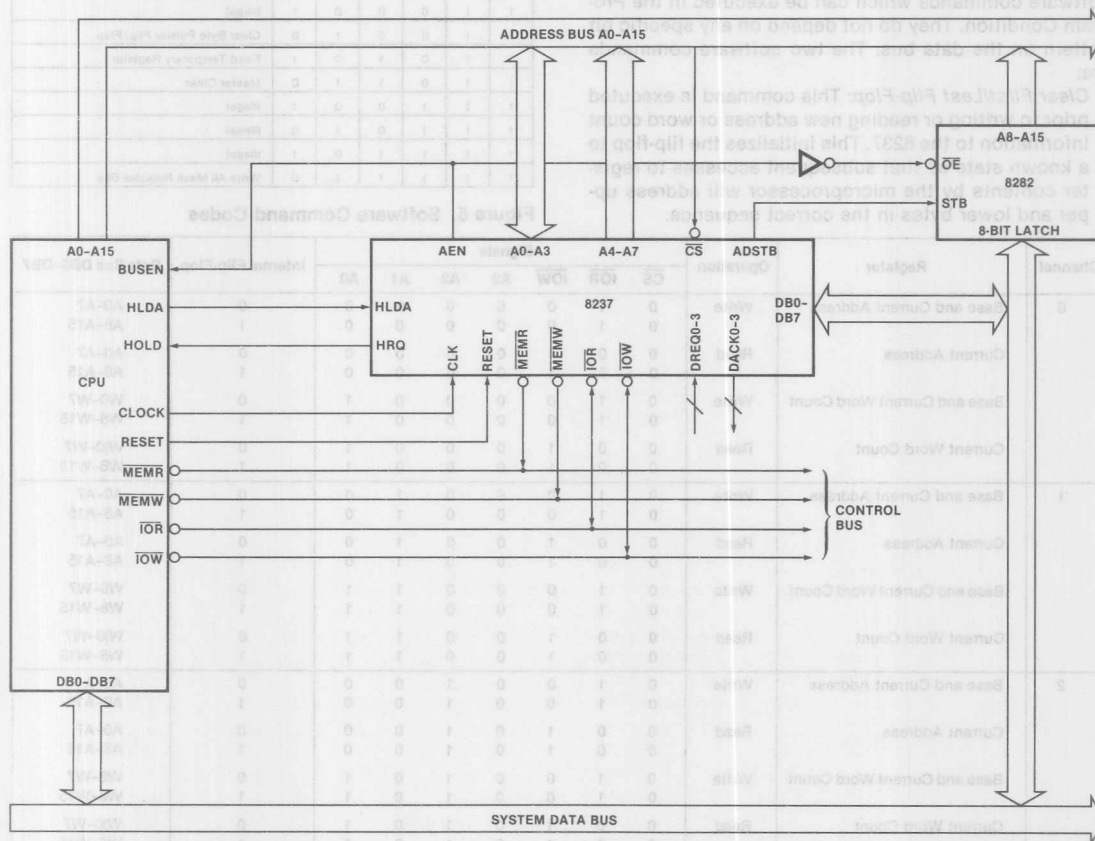


Figure 7. 8237 System Interface



**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature under Bias . . . . . 0°C to 70°C  
 Storage Temperature . . . . . - 65°C to + 150°C  
 Voltage on any Pin with Respect to Ground . . . . . - 0.5 to 7V  
 Power Dissipation . . . . . 1.5 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

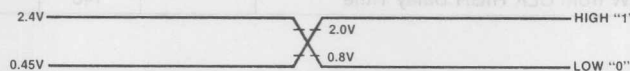
**D.C. CHARACTERISTICS**

$T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 5\%$ ,  $GND = 0\text{V}$

Symbol	Parameter	Min.	Typ. <sup>(1)</sup>	Max.	Unit	Test Conditions
$V_{OH}$	Output HIGH Voltage	2.4			V	$I_{OH} = -200\ \mu\text{A}$
		3.3			V	$I_{OH} = -100\ \mu\text{A}$ (HRQ Only)
$V_{OL}$	Output LOW Voltage			0.4	V	$I_{OL} = 3.2\ \text{mA}$
$V_{IH}$	Input HIGH Voltage	2.0		$V_{CC} + 0.5$	V	
$V_{IL}$	Input LOW Voltage	- 0.5		0.8	V	
$I_{LI}$	Input Load Current			$\pm 10$	$\mu\text{A}$	$V_{SS} \leq V_I \leq V_{CC}$
$I_{LO}$	Output Leakage Current			$\pm 10$	$\mu\text{A}$	$V_{CC} \leq V_O \leq V_{SS} + 0.40$
$I_{CC}$	$V_{CC}$ Supply Current		65	130	mA	$T_A = +25^\circ\text{C}$
			75	150	mA	$T_A = 0^\circ\text{C}$
$C_O$	Output Capacitance		4	8	pF	$f_c = 1.0\ \text{MHz}$ , Inputs = 0V
$C_I$	Input Capacitance		8	15	pF	
$C_{IO}$	I/O Capacitance		10	18	pF	

**Notes:**

- Typical values are for  $T_A = 25^\circ\text{C}$ , nominal supply voltage and nominal processing parameters.
- Input timing parameters assume transition times of 20 ns or less. Waveform measurement points for both input and output signals are 2.0V for HIGH and 0.8V for LOW, unless otherwise noted.
- Output loading is 1 TTL gate plus 50 pF capacitance, unless otherwise noted.
- The net  $I_{OW}$  or  $MEMW$  Pulse width for normal write will be  $TCY-100$  ns and for extended write will be  $2TCY-100$  ns. The net  $I_{OR}$  or  $MEMR$  pulse width for normal read will be  $2TCY-50$  ns and for compressed read will be  $TCY-50$  ns.
- TDQ is specified for two different output HIGH levels. TDQ1 is measured at 2.0V. TDQ2 is measured at 3.3V. The value for TDQ2 assumes an external 3.3 k $\Omega$  pull-up resistor connected from HRQ to  $V_{CC}$ .
- DREQ should be held active until DACK is returned.
- DREQ and DACK signals may be active high or active low. Timing diagrams assume the active high mode.
- Output loading on the data bus is 1 TTL gate plus 100 pF capacitance.
- Successive read and/or write operations by the external processor to program or examine the controller must be timed to allow at least 600 ns for the 8237 and at least 400 ns for the 8237-2 as recovery time between active read or write pulses.
- Parameters are listed in alphabetical order.
- Pin 5 is an input that should always be at a logic high level. An internal pull-up resistor will establish a logic high when the pin is left floating. Alternatively, pin 5 may be tied to  $V_{CC}$ .

**A.C. TEST WAVEFORM**

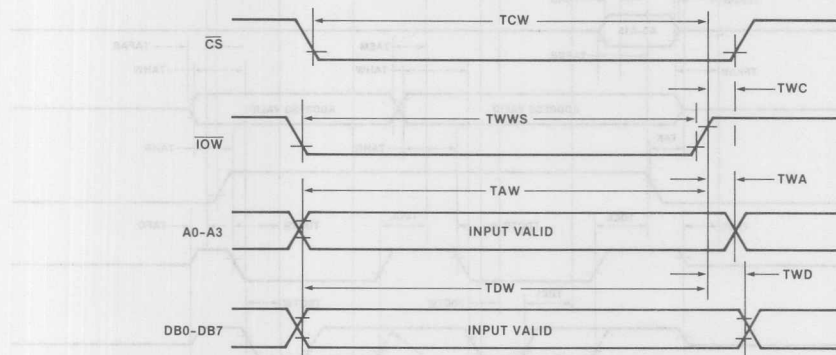
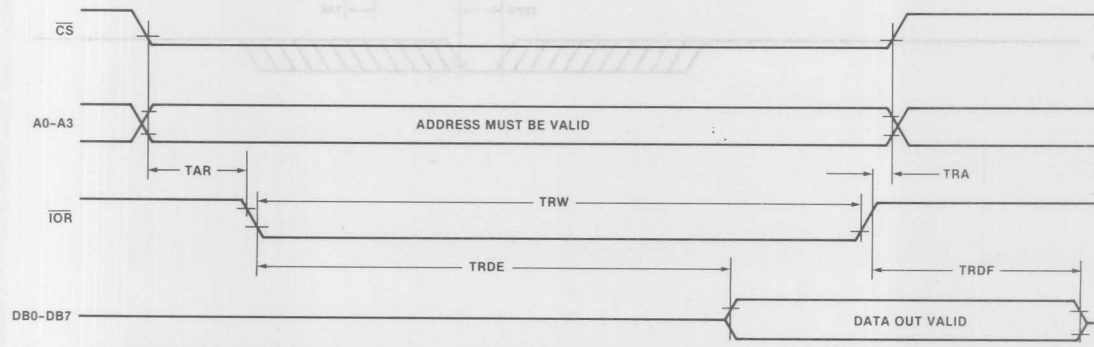
**A.C. CHARACTERISTICS: DMA (MASTER) MODE**T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = 5.0V ± 5%, GND = 0V

Symbol	Parameter	8237		8237-2		Unit
		Min.	Max.	Min.	Max.	
TAEL	AEN HIGH from CLK LOW (S1) Delay Time		300		200	ns
TAET	AEN LOW from CLK HIGH (S1) Delay Time		200		130	ns
TAFAB	ADR Active to Float Delay from CLK HIGH		150		90	ns
TAFC	READ or WRITE Float from CLK HIGH		150		120	ns
TAFDB	DB Active to Float Delay from CLK HIGH		250		170	ns
TAHR	ADR from READ HIGH Hold Time	TCY-100		TCY-100		ns
TAHS	DB from ADSTB LOW Hold Time	50		30		ns
TAHW	ADR from WRITE HIGH Hold Time	TCY-50		TCY-50		ns
TAK	DACK Valid from CLK LOW Delay Time		250		170	ns
	EOP HIGH from CLK HIGH Delay Time		250		170	ns
	EOP LOW to CLK HIGH Delay Time		250		100	ns
TASM	ADR Stable from CLK HIGH		250		170	ns
TASS	DB to ADSTB LOW Setup Time	100		100		ns
TCH	Clock High Time (Transitions ≤ 10 ns)	120		70		ns
TCL	Clock LOW Time (Transitions ≤ 10 ns)	150		50		ns
TCY	CLK Cycle Time	320		200		ns
TDCL	CLK HIGH to READ or WRITE LOW Delay (Note 4)		270		190	ns
TDCTR	READ HIGH from CLK HIGH (S4) Delay Time (Note 4)		270		190	ns
TDCTW	WRITE HIGH from CLK HIGH (S4) Delay Time (Note 4)		200		130	ns
TDQ1	HRQ Valid from CLK HIGH Delay Time (Note 5)		160		120	ns
TDQ2			250		120	ns
TEPS	EOP LOW from CLK LOW Setup Time	60		40		ns
TEPW	EOP Pulse Width	300		220		ns
TFAAB	ADR Float to Active Delay from CLK HIGH		250		170	ns
TFAC	READ or WRITE Active from CLK HIGH		200		150	ns
TFADB	DB Float to Active Delay from CLK HIGH		300		200	ns
THS	HCDA Valid to CLK HIGH Setup Time	100		75		ns
TIDH	Input Data from MEMR HIGH Hold Time	0		0		ns
TIDS	Input Data to MEMR HIGH Setup Time	250		170		ns
TODH	Output Data from MEMW HIGH Hold Time	20		10		ns
TODV	Output Data Valid to MEMW HIGH	200		130		ns
TQS	DREQ to CLK LOW (S1, S4) Setup Time	0		0		ns
TRH	CLK to READY LOW Hold Time	20		20		ns
TRS	READY to CLK LOW Setup Time	100		75		ns
TSTL	ADSTB HIGH from CLK HIGH Delay Time		200		130	ns
TSTT	ADSTB LOW from CLK HIGH Delay Time		140		90	ns

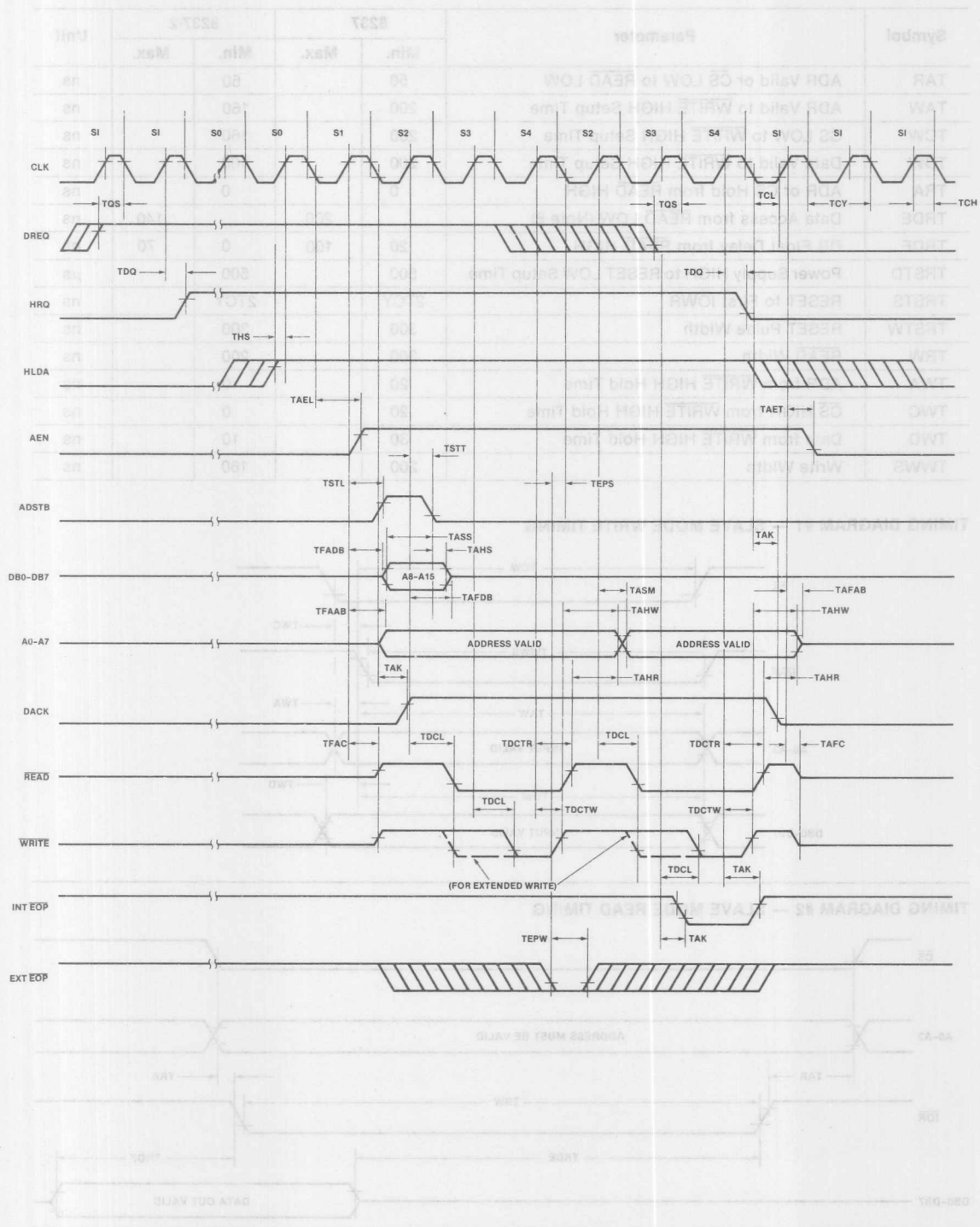
**A.C. CHARACTERISTICS: PERIPHERAL (SLAVE) MODE**

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 5\%$ ,  $GND = 0\text{V}$

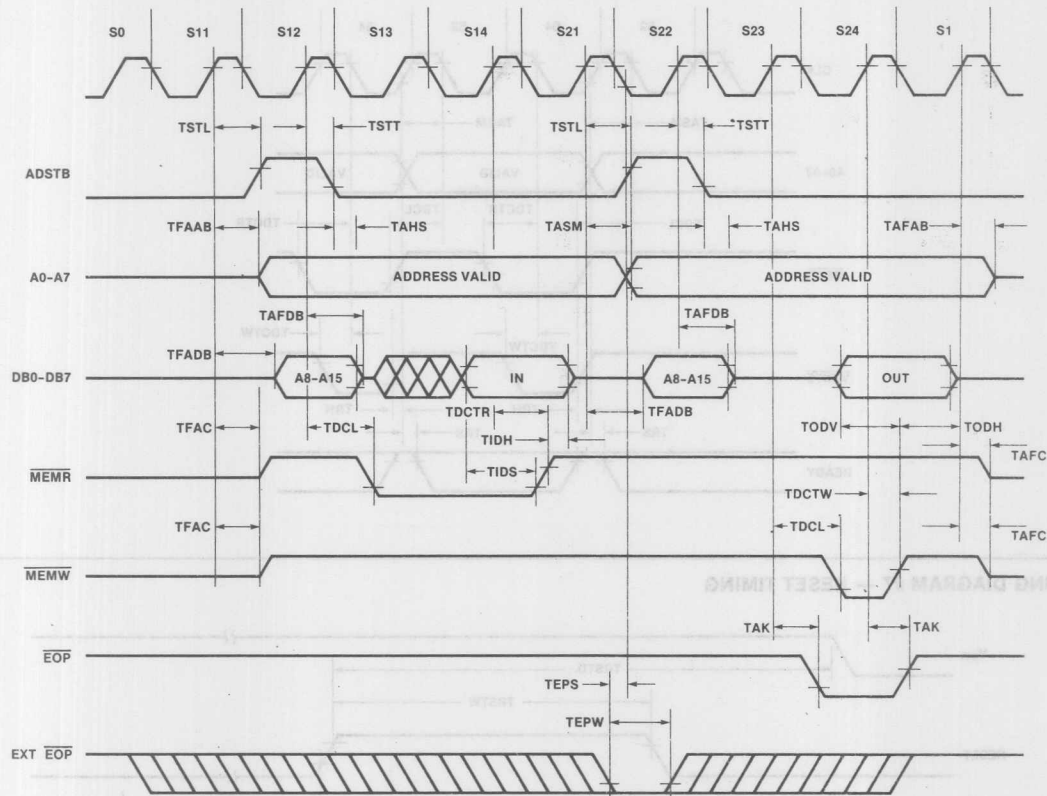
Symbol	Parameter	8237		8237-2		Unit
		Min.	Max.	Min.	Max.	
TAR	ADR Valid or $\overline{CS}$ LOW to $\overline{READ}$ LOW	50		50		ns
TAW	ADR Valid to $\overline{WRITE}$ HIGH Setup Time	200		160		ns
TCW	CS LOW to $\overline{WRITE}$ HIGH Setup Time	200		160		ns
TDW	Data Valid to $\overline{WRITE}$ HIGH Setup Time	200		160		ns
TRA	ADR or CS Hold from $\overline{READ}$ HIGH	0		0		ns
TRDE	Data Access from $\overline{READ}$ LOW (Note 8)		200		140	ns
TRDF	DB Float Delay from $\overline{READ}$ HIGH	20	100	0	70	ns
TRSTD	Power Supply HIGH to RESET LOW Setup Time	500		500		$\mu\text{s}$
TRSTS	RESET to First $\overline{IOWR}$	2TCY		2TCY		ns
TRSTW	RESET Pulse Width	300		300		ns
TRW	$\overline{READ}$ Width	300		200		ns
TWA	ADR from $\overline{WRITE}$ HIGH Hold Time	20		0		ns
TWC	$\overline{CS}$ HIGH from $\overline{WRITE}$ HIGH Hold Time	20		0		ns
TWD	Data from $\overline{WRITE}$ HIGH Hold Time	30		10		ns
TWWS	Write Width	200		160		ns

**TIMING DIAGRAM #1 — SLAVE MODE WRITE TIMING****TIMING DIAGRAM #2 — SLAVE MODE READ TIMING**

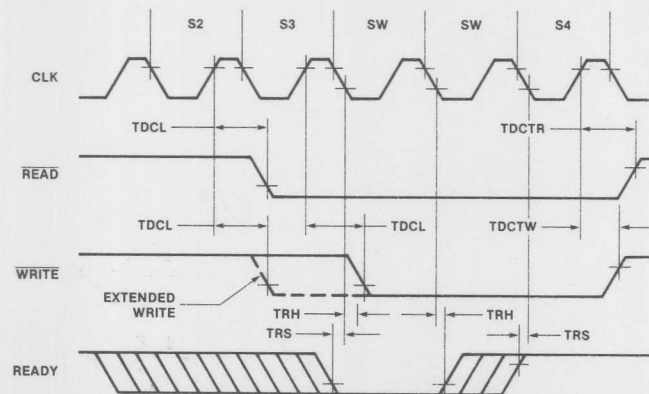
TIMING DIAGRAM #3 — DMA TRANSFER TIMING



TIMING DIAGRAM #4 — MEMORY TO MEMORY TRANSFER TIMING

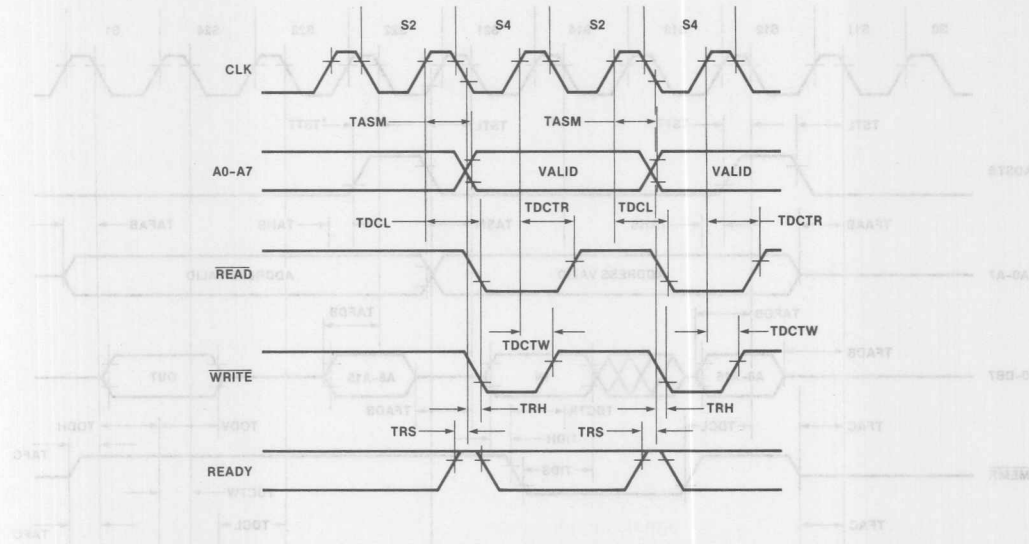


TIMING DIAGRAM #5 — READY TIMING

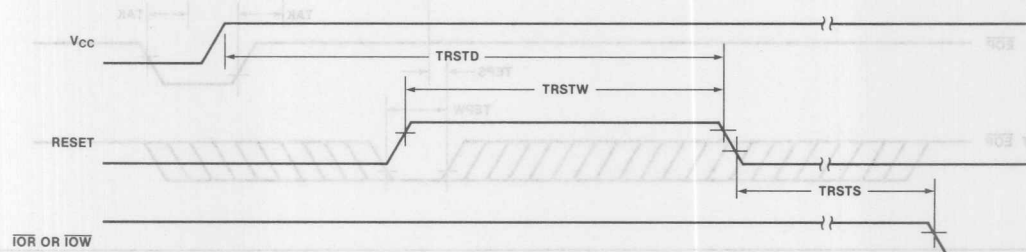




TIMING DIAGRAM #6 — COMPRESSED TRANSFER TIMING



TIMING DIAGRAM #7 — RESET TIMING



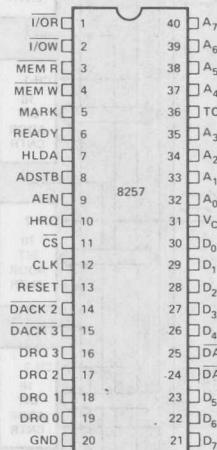
# 8257/8257-5 PROGRAMMABLE DMA CONTROLLER

- MCS-85™ Compatible 8257-5
- 4-Channel DMA Controller
- Priority DMA Request Logic
- Channel Inhibit Logic

- Terminal Count and Modulo 128 Outputs
- Single TTL Clock
- Single +5V Supply
- Auto Load Mode

The Intel® 8257 is a 4-channel direct memory access (DMA) controller. It is specifically designed to simplify the transfer of data at high speeds for the Intel® microcomputer systems. Its primary function is to generate, upon a peripheral request, a sequential memory address which will allow the peripheral to read or write data directly to or from memory. Acquisition of the system bus is accomplished via the CPU's hold function. The 8257 has priority logic that resolves the peripherals requests and issues a composite hold request to the CPU. It maintains the DMA cycle count for each channel and outputs a control signal to notify the peripheral that the programmed number of DMA cycles is complete. Other output control signals simplify sectorized data transfers. The 8257 represents a significant savings in component count for DMA-based microcomputer systems and greatly simplifies the transfer of data at high speed between peripherals and memories.

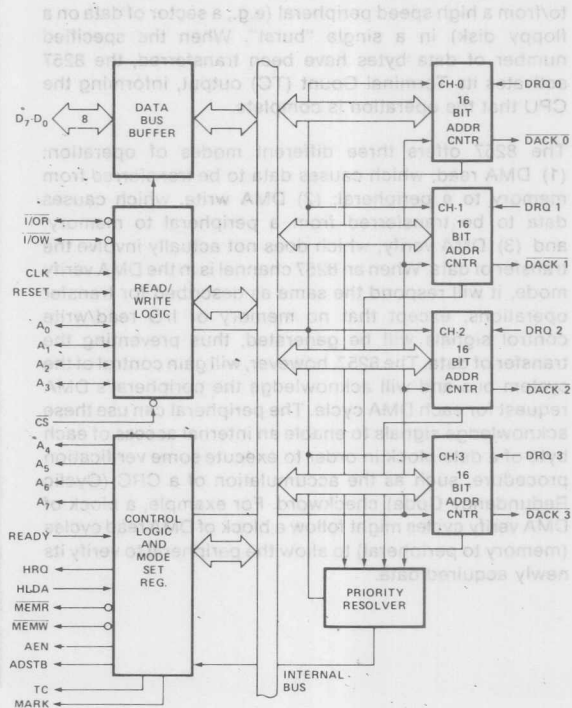
## PIN CONFIGURATION



## PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	DATA BUS	AEN	ADDRESS ENABLE
A <sub>7</sub> -A <sub>0</sub>	ADDRESS BUS	ADSTB	ADDRESS STROBE
I/OR	I/O READ	TC	TERMINAL COUNT
I/OW	I/O WRITE	MARK	MODULO 128 MARK
MEMR	MEMORY READ	DRQ <sub>3</sub> -DRQ <sub>0</sub>	DMA REQUEST INPUT
MEMW	MEMORY WRITE	DACK <sub>3</sub> -DACK <sub>0</sub>	DMA ACKNOWLEDGE OUT
CLK	CLOCK INPUT	CS	CHIP SELECT
RESET	RESET INPUT	V <sub>cc</sub>	+5 VOLTS
READY	READY	GND	GROUND
HRQ	HOLD REQUEST (TO 8080A)		
HLDA	HOLD ACKNOWLEDGE (FROM 8080A)		

## BLOCK DIAGRAM



## General

The 8257 is a programmable, Direct Memory Access (DMA) device which, when coupled with a single Intel® 8212 I/O port device, provides a complete four-channel DMA controller for use in Intel® microcomputer systems. After being initialized by software, the 8257 can transfer a block of data, containing up to 16,384 bytes, between memory and a peripheral device directly, without further intervention required of the CPU. Upon receiving a DMA transfer request from an enabled peripheral, the 8257:

1. Acquires control of the system bus.
2. Acknowledges that requesting peripheral which is connected to the highest priority channel.
3. Outputs the least significant eight bits of the memory address onto system address lines  $A_0-A_7$ , outputs the most significant eight bits of the memory address to the 8212 I/O port via the data bus (the 8212 places these address bits on lines  $A_8-A_{15}$ ), and
4. Generates the appropriate memory and I/O read/write control signals that cause the peripheral to receive or deposit a data byte directly from or to the addressed location in memory.

The 8257 will retain control of the system bus and repeat the transfer sequence, as long as a peripheral maintains its DMA request. Thus, the 8257 can transfer a block of data to/from a high speed peripheral (e.g., a sector of data on a floppy disk) in a single "burst". When the specified number of data bytes have been transferred, the 8257 activates its Terminal Count (TC) output, informing the CPU that the operation is complete.

The 8257 offers three different modes of operation: (1) DMA read, which causes data to be transferred from memory to a peripheral; (2) DMA write, which causes data to be transferred from a peripheral to memory; and (3) DMA verify, which does not actually involve the transfer of data. When an 8257 channel is in the DMA verify mode, it will respond the same as described for transfer operations, except that no memory or I/O read/write control signals will be generated, thus preventing the transfer of data. The 8257, however, will gain control of the system bus and will acknowledge the peripheral's DMA request for each DMA cycle. The peripheral can use these acknowledge signals to enable an internal access of each byte of a data block in order to execute some verification procedure, such as the accumulation of a CRC (Cyclic Redundancy Code) checkword. For example, a block of DMA verify cycles might follow a block of DMA read cycles (memory to peripheral) to allow the peripheral to verify its newly acquired data.

## Block Diagram Description

### 1. DMA Channels

The 8257 provides four separate DMA channels (labeled CH-0 to CH-3). Each channel includes two sixteen-bit registers: (1) a DMA address register, and (2) a terminal count register. Both registers must be initialized before a channel is enabled. The DMA address register is loaded with the address of the first memory location to be accessed. The value loaded into the low-order 14-bits of the terminal count register specifies the number of DMA cycles minus one before the Terminal Count (TC) output is activated. For instance, a terminal count of 0 would cause the TC output to be active in the first DMA cycle for that channel. In general, if  $N$  = the number of desired DMA cycles, load the value  $N-1$  into the low-order 14-bits of the terminal count register. The most significant two bits of the terminal count register specify the type of DMA operation for that channel.

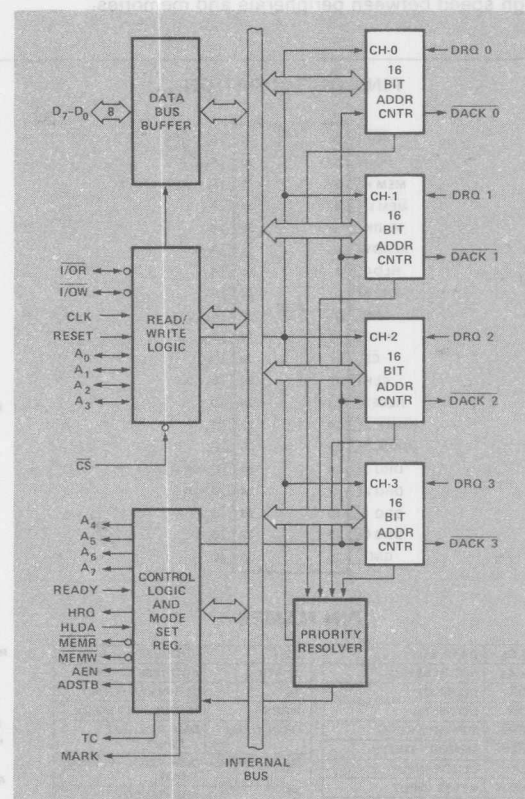


Figure 1. 8257 Block Diagram Showing DMA Channels

These two bits are not modified during a DMA cycle, but can be changed between DMA blocks.

Each channel accepts a DMA Request (DRQn) input and provides a DMA Acknowledge (DACKn) output.

### (DRQ 0-DRQ 3)

**DMA Request:** These are individual asynchronous channel request inputs used by the peripherals to obtain a DMA cycle. If not in the rotating priority mode then DRQ 0 has the highest priority and DRQ 3 has the lowest. A request can be generated by raising the request line and holding it high until DMA acknowledge. For multiple DMA cycles (Burst Mode) the request line is held high until the DMA acknowledge of the last cycle arrives.

### (DACK 0 - DACK 3)

**DMA Acknowledge:** An active low level on the acknowledge output informs the peripheral connected to that channel that it has been selected for a DMA cycle. The DACK output acts as a "chip select" for the peripheral device requesting service. This line goes active (low) and inactive (high) once for each byte transferred even if a burst of data is being transferred.

## 2. Data Bus Buffer

This three-state, bi-directional, eight bit buffer interfaces the 8257 to the system data bus.

### (D<sub>0</sub>-D<sub>7</sub>)

**Data Bus Lines:** These are bi-directional three-state lines. When the 8257 is being programmed by the CPU, eight-bits of data for a DMA address register, a terminal count register or the Mode Set register are received on the data bus. When the CPU reads a DMA address register, a terminal count register or the Status register, the data is sent to the CPU over the data bus. During DMA cycles (when the 8257 is the bus master), the 8257 will output the most significant eight-bits of the memory address (from one of the DMA address registers) to the 8212 latch via the data bus. These address bits will be transferred at the beginning of the DMA cycle; the bus will then be released to handle the memory data transfer during the balance of the DMA cycle.

BIT 15	BIT 14	TYPE OF DMA OPERATION
0	0	Verify DMA Cycle
0	1	Write DMA Cycle
1	0	Read DMA Cycle
1	1	(Illegal)

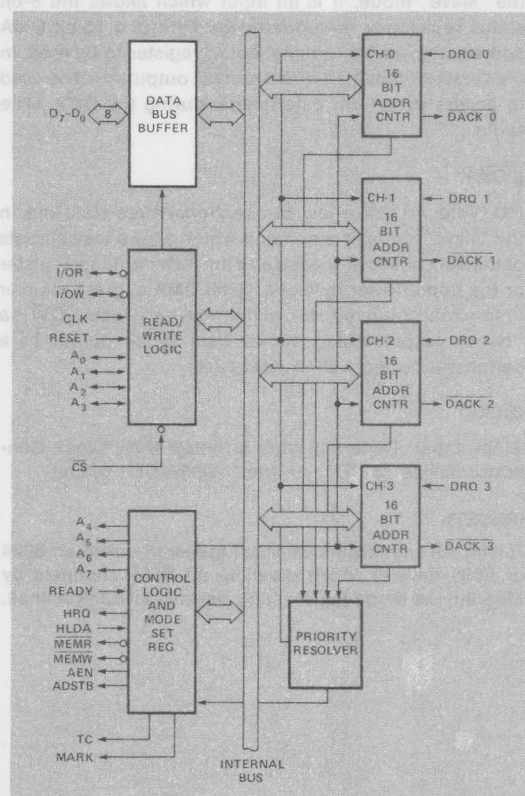


Figure 2. 8257 Block Diagram Showing Data Bus Buffer

### 3. Read/Write Logic

When the CPU is programming or reading one of the 8257's registers (i.e., when the 8257 is a "slave" device on the system bus), the Read/Write Logic accepts the I/O Read ( $\overline{I/O\overline{R}}$ ) or I/O Write ( $\overline{I/O\overline{W}}$ ) signal, decodes the least significant four address bits, ( $A_0-A_3$ ), and either writes the contents of the data bus into the addressed register (if  $\overline{I/O\overline{W}}$  is true) or places the contents of the addressed register onto the data bus (if  $\overline{I/O\overline{R}}$  is true).

During DMA cycles (i.e., when the 8257 is the bus "master"), the Read/Write Logic generates the I/O read and memory write (DMA write cycle) or I/O Write and memory read (DMA read cycle) signals which control the data link with the peripheral that has been granted the DMA cycle.

Note that during DMA transfers Non-DMA I/O devices should be de-selected (disabled) using "AEN" signal to inhibit I/O device decoding of the memory address as an erroneous device address.

#### ( $\overline{I/O\overline{R}}$ )

I/O Read: An active-low, bi-directional three-state line. In the "slave" mode, it is an input which allows the 8-bit status register or the upper/lower byte of a 16-bit DMA address register or terminal count register to be read. In the "master" mode,  $\overline{I/O\overline{R}}$  is a control output which is used to access data from a peripheral during the DMA write cycle.

#### ( $\overline{I/O\overline{W}}$ )

I/O Write: An active-low, bi-directional three-state line. In the "slave" mode, it is an input which allows the contents of the data bus to be loaded into the 8-bit mode set register or the upper/lower byte of a 16-bit DMA address register or terminal count register. In the "master" mode,  $\overline{I/O\overline{W}}$  is a control output which allows data to be output to a peripheral during a DMA read cycle.

#### (CLK)

Clock Input: Generally from an Intel® 8224 Clock Generator device. ( $\phi$ 2 TTL) or Intel® 8085A CLK output.

#### (RESET)

Reset: An asynchronous input (generally from an 8224 or 8085 device) which disables all DMA channels by clearing the mode register and 3-states all control lines.

#### ( $A_0-A_3$ )

Address Lines: These least significant four address lines are bi-directional. In the "slave" mode they are inputs which select one of the registers to be read or programmed. In the "master" mode, they are outputs which constitute the least significant four bits of the 16-bit memory address generated by the 8257.

#### (CS)

Chip Select: An active-low input which enables the I/O Read or I/O Write input when the 8257 is being read or programmed in the "slave" mode. In the "master" mode, CS is automatically disabled to prevent the chip from selecting itself while performing the DMA function.

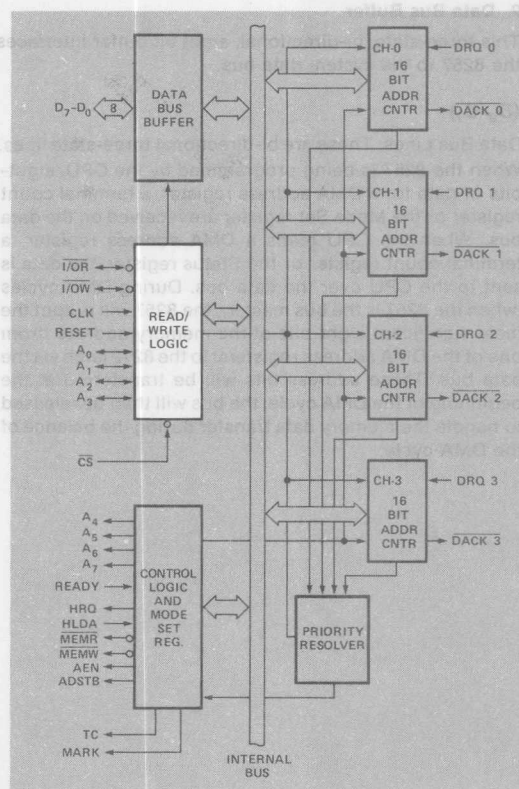


Figure 3. 8257 Block Diagram Showing Read/Write Logic Function



#### 4. Control Logic

This block controls the sequence of operations during all DMA cycles by generating the appropriate control signals and the 16-bit address that specifies the memory location to be accessed.

##### (A<sub>4</sub>-A<sub>7</sub>)

**Address Lines:** These four address lines are three-state outputs which constitute bits 4 through 7 of the 16-bit memory address generated by the 8257 during all DMA cycles.

##### (READY)

**Ready:** This asynchronous input is used to elongate the memory read and write cycles in the 8257 with wait states, if the selected memory requires longer cycles.

##### (HRQ)

**Hold Request:** This output requests control of the system bus. In systems with only one 8257, HRQ will normally be applied to the HOLD input on the CPU.

##### (HLDA)

**Hold Acknowledge:** This input from the CPU indicates that the 8257 has acquired control of the system bus.

##### (MEMR)

**Memory Read:** This active-low three-state output is used to read data from the addressed memory location during DMA Read cycles.

##### (MEMW)

**Memory Write:** This active-low three-state output is used to write data into the addressed memory location during DMA Write cycles.

##### (ADSTB)

**Address Strobe:** This output strobes the most significant byte of the memory address into the 8212 device from the data bus.

##### (AEN)

**Address Enable:** This output is used to disable (float) the System Data Bus and the System Control Bus. It may also be used to disable (float) the System Address Bus by use of an enable on the Address Bus drivers in systems to inhibit non-DMA devices from responding during DMA cycles. It may be further used to isolate the 8257 data bus from the System Data Bus to facilitate the transfer of the 8 most significant DMA address bits over the 8257 data I/O pins without subjecting the System Data Bus to any timing constraints for the transfer. When the 8257 is used in an I/O device structure (as opposed to memory mapped), this AEN output should be used to disable the selection of an I/O device when the DMA address is on the address bus. The I/O device selection should be determined by the DMA acknowledge outputs for the 4 channels.

##### (TC)

**Terminal Count:** This output notifies the currently selected peripheral that the present DMA cycle should be the last cycle for this data block. If the TC STOP bit in the Mode Set register is set, the selected channel will be automatically disabled at the end of that DMA cycle. TC is activated when the 14-bit value in the selected channel's terminal count register equals zero. Recall that the low-order 14-bits of the terminal count register should be loaded with the values (n-1), where n = the desired number of the DMA cycles.

##### (MARK)

**Modulo 128 Mark:** This output notifies the selected peripheral that the current DMA cycle is the 128th cycle since the previous MARK output. MARK always occurs at 128 (and all multiples of 128) cycles from the end of the data block. Only if the total number of DMA cycles (n) is evenly divisible by 128 (and the terminal count register was loaded with n-1), will MARK occur at 128 (and each succeeding multiple of 128) cycles from the beginning of the data block.

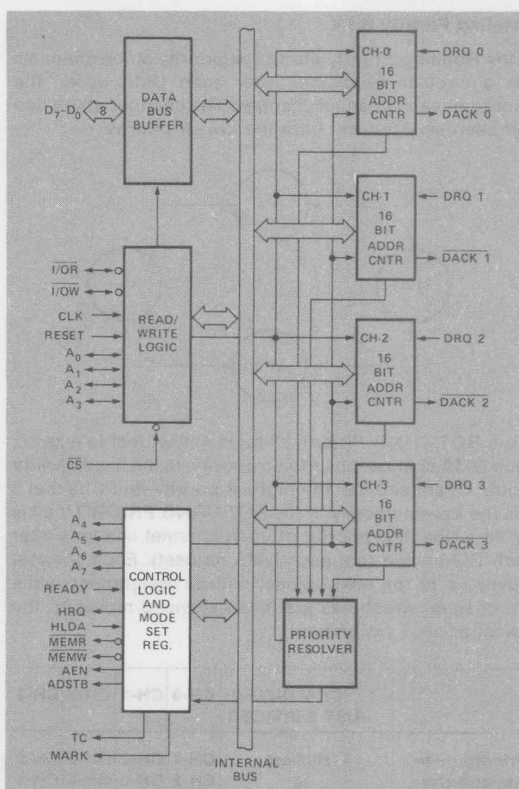
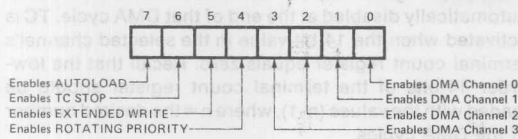


Figure 4. 8257 Block Diagram Showing Control Logic and Mode Set Register

### 5. Mode Set Register

When set, the various bits in the Mode Set register enable each of the four DMA channels, and allow four different options for the 8257:

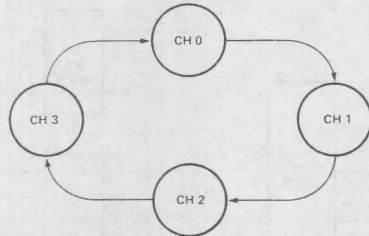


The Mode Set register is normally programmed by the CPU after the DMA address register(s) and terminal count register(s) are initialized. The Mode Set Register is cleared by the RESET input, thus disabling all options, inhibiting all channels, and preventing bus conflicts on power-up. A channel should not be left enabled unless its DMA address and terminal count registers contain valid values; otherwise, an inadvertent DMA request (DRQn) from a peripheral could initiate a DMA cycle that would destroy memory data.

The various options which can be enabled by bits in the Mode Set register are explained below:

#### Rotating Priority Bit 4

In the Rotating Priority Mode, the priority of the channels has a circular sequence. After each DMA cycle, the priority of each channel changes. The channel which has just been serviced will have the lowest priority.



If the ROTATING PRIORITY bit is not set (set to a zero), each DMA channel has a fixed priority. In the fixed priority mode, Channel 0 has the highest priority and Channel 3 has the lowest priority. If the ROTATING PRIORITY bit is set to a one, the priority of each channel changes after each DMA cycle (not each DMA request). Each channel moves up to the next highest priority assignment, while the channel which has just been serviced moves to the lowest priority assignment:

	CHANNEL → JUST SERVICED	CH-0	CH-1	CH-2	CH-3
Priority → Assignments	Highest ↑ Lowest	CH-1 CH-2 CH-3 CH-0	CH-2 CH-3 CH-0 CH-1	CH-3 CH-0 CH-1 CH-2	CH-0 CH-1 CH-2 CH-3

Note that rotating priority will prevent any one channel from monopolizing the DMA mode; consecutive DMA cycles will service different channels if more than one channel is enabled and requesting service. There is no overhead penalty associated with this mode of operation. All DMA operations began with Channel 0 initially assigned to the highest priority for the first DMA cycle.

#### Extended Write Bit 5

If the EXTENDED WRITE bit is set, the duration of both the MEMW and I/OW signals is extended by activating them earlier in the DMA cycle. Data transfers within micro-computer systems proceed asynchronously to allow use of various types of memory and I/O devices with different access times. If a device cannot be accessed within a specific amount of time it returns a "not ready" indication to the 8257 that causes the 8257 to insert one or more wait states in its internal sequencing. Some devices are fast enough to be accessed without the use of wait states, but if they generate their READY response with the leading edge of the I/OW or MEMW signal (which generally occurs late in the transfer sequence), they would normally cause the 8257 to enter a wait state because it does not receive READY in time. For systems with these types of devices, the Extended Write option provides alternative timing for the I/O and memory write signals which allows the devices to return an early READY and prevents the unnecessary occurrence of wait states in the 8257, thus increasing system throughput.

#### TC Stop Bit 6

If the TC STOP bit is set, a channel is disabled (i.e., its enable bit is reset) after the Terminal Count (TC) output goes true, thus automatically preventing further DMA operation on that channel. The enable bit for that channel must be re-programmed to continue or begin another DMA operation. If the TC STOP bit is not set, the occurrence of the TC output has no effect on the channel enable bits. In this case, it is generally the responsibility of the peripheral to cease DMA requests in order to terminate a DMA operation.

#### Auto Load Bit 7

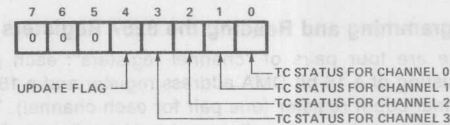
The Auto Load mode permits Channel 2 to be used for repeat block or block chaining operations, without immediate software intervention between blocks. Channel 2 registers are initialized as usual for the first data block; Channel 3 registers, however, are used to store the block re-initialization parameters (DMA starting address, terminal count and DMA transfer mode). After the first block of DMA cycles is executed by Channel 2 (i.e., after the TC output goes true), the parameters stored in the Channel 3 registers are transferred to Channel 2 during an "update" cycle. Note that the TC STOP feature, described above, has no effect on Channel 2 when the Auto Load bit is set.

If the Auto Load bit is set, the initial parameters for Channel 2 are automatically duplicated in the Channel 3 registers when Channel 2 is programmed. This permits repeat block operations to be set up with the programming of a single channel. Repeat block operations can be used in applications such as CRT refreshing. Channels 2 and 3 can still be loaded with separate values if Channel 2 is loaded before loading Channel 3. Note that in the Auto Load mode, Channel 3 is still available to the user if the Channel 3 enable bit is set, but use of this channel will change the values to be auto loaded into Channel 2 at update time. All that is necessary to use the Auto Load feature for chaining operations is to reload Channel 3 registers at the conclusion of each update cycle with the new parameters for the next data block transfer.

Each time that the 8257 enters an update cycle, the update flag in the status register is set and parameters in Channel 3 are transferred to Channel 2, non-destructively for Channel 3. The actual re-initialization of Channel 2 occurs at the beginning of the next channel 2 DMA cycle after the TC cycle. This will be the first DMA cycle of the new data block for Channel 2. The update flag is cleared at the conclusion of this DMA cycle. For chaining operations, the update flag in the status register can be monitored by the CPU to determine when the re-initialization process has been completed so that the next block parameters can be safely loaded into Channel 3.

## 6. Status Register

The eight-bit status register indicates which channels have reached a terminal count condition and includes the update flag described previously.



The TC status bits are set when the Terminal Count (TC) output is activated for that channel. These bits remain set until the status register is read or the 8257 is reset. The UPDATE FLAG, however, is not affected by a status register read operation. The UPDATE FLAG can be cleared by resetting the 8257, by changing to the non-auto load mode (i.e., by resetting the AUTO LOAD bit in the Mode Set register) or it can be left to clear itself at the completion of the update cycle. The purpose of the UPDATE FLAG is to prevent the CPU from inadvertently skipping a data block by overwriting a starting address or terminal count in the Channel 3 registers before those parameters are properly auto-loaded into Channel 2.

The user is cautioned against reading the TC status register and using this information to reenable channels that have not completed operation. Unless the DMA channels are inhibited a channel could reach terminal count (TC) between the status read and the mode write. DMA can be inhibited by a hardware gate on the HRQ line or by disabling channels with a mode word before reading the TC status.

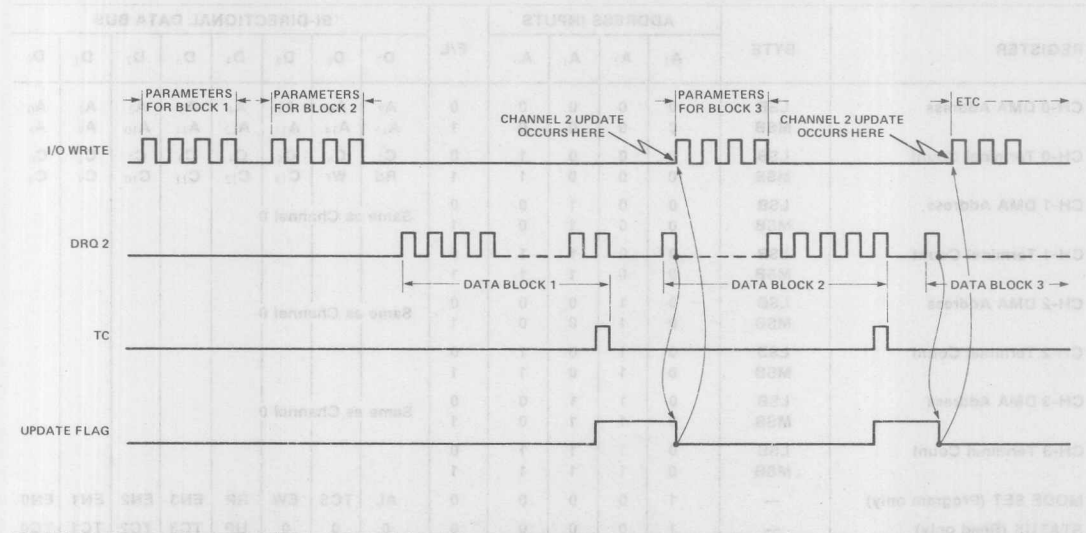


Figure 5. Autoload Timing

## OPERATIONAL SUMMARY

### Programming and Reading the 8257 Registers

There are four pairs of "channel registers": each pair consisting of a 16-bit DMA address register and a 16-bit terminal count register (one pair for each channel). The 8257 also includes two "general registers": one 8-bit Mode Set register and one 8-bit Status register. The registers are loaded or read when the CPU executes a write or read instruction that addresses the 8257 device and the appropriate register within the 8257. The 8228 generates the appropriate read or write control signal (generally I/OR or I/OW while the CPU places a 16-bit address on the system address bus, and either outputs the data to be written onto the system data bus or accepts the data being read from the data bus. All or some of the most significant 12 address bits  $A_4$ - $A_{15}$  (depending on the systems memory, I/O configuration) are usually decoded to produce the chip select ( $\overline{CS}$ ) input to the 8257. An I/O Write input (or Memory Write in memory mapped I/O configurations, described below) specifies that the addressed register is to be programmed, while an I/O Read input (or Memory Read) specifies that the addressed register is to be read. Address bit 3 specifies whether a "channel register" ( $A_3 = 0$ ) or the Mode Set (program only)/Status (read only) register ( $A_3 = 1$ ) is to be accessed.

The least significant three address bits,  $A_0$ - $A_2$ , indicate the specific register to be accessed. When accessing the Mode Set or Status register,  $A_0$ - $A_2$  are all zero. When accessing a channel register bit  $A_0$  differentiates between the DMA address register ( $A_0 = 0$ ) and the terminal count register ( $A_0 = 1$ ), while bits  $A_1$  and  $A_2$  specify one of the

CONTROL INPUT	$\overline{CS}$	I/OW	I/OR	$A_3$
Program Half of a Channel Register	0	0	1	0
Read Half of a Channel Register	0	1	0	0
Program Mode Set Register	0	0	1	1
Read Status Register	0	1	0	1

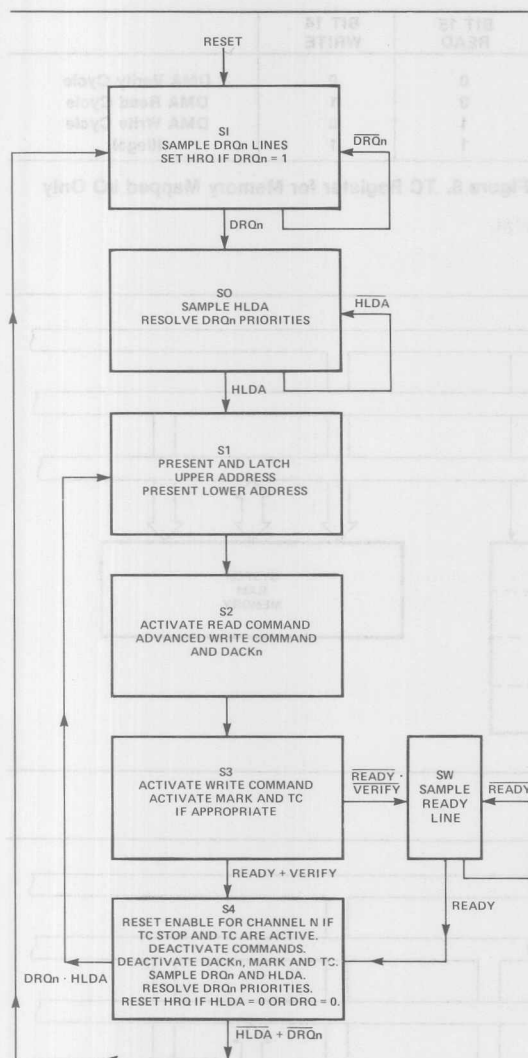
four channels. Because the "channel registers" are 16-bits, two program instruction cycles are required to load or read an entire register. The 8257 contains a first/last (F/L) flip flop which toggles at the completion of each channel program or read operation. The F/L flip flop determines whether the upper or lower byte of the register is to be accessed. The F/L flip flop is reset by the RESET input and whenever the Mode Set register is loaded. To maintain proper synchronization when accessing the "channel registers" all channel command instruction operations should occur in pairs, with the lower byte of a register always being accessed first. Do not allow  $\overline{CS}$  to clock while either I/OR or I/OW is active, as this will cause an erroneous F/L flip flop state. In systems utilizing an interrupt structure, interrupts should be disabled prior to any paired programming operations to prevent an interrupt from splitting them. The result of such a split would leave the F/L F/F in the wrong state. This problem is particularly obvious when other DMA channels are programmed by an interrupt structure.

### 8257 Register Selection

REGISTER	BYTE	ADDRESS INPUTS				F/L	*BI-DIRECTIONAL DATA BUS							
		$A_3$	$A_2$	$A_1$	$A_0$		$D_7$	$D_6$	$D_5$	$D_4$	$D_3$	$D_2$	$D_1$	$D_0$
CH-0 DMA Address	LSB	0	0	0	0	0	$A_7$	$A_6$	$A_5$	$A_4$	$A_3$	$A_2$	$A_1$	$A_0$
	MSB	0	0	0	0	1	$A_{15}$	$A_{14}$	$A_{13}$	$A_{12}$	$A_{11}$	$A_{10}$	$A_9$	$A_8$
CH-0 Terminal Count	LSB	0	0	0	1	0	$C_7$	$C_6$	$C_5$	$C_4$	$C_3$	$C_2$	$C_1$	$C_0$
	MSB	0	0	0	1	1	Rd	Wr	$C_{13}$	$C_{12}$	$C_{11}$	$C_{10}$	$C_9$	$C_8$
CH-1 DMA Address	LSB	0	0	1	0	0	Same as Channel 0							
	MSB	0	0	1	0	1								
CH-1 Terminal Count	LSB	0	0	1	1	0	Same as Channel 0							
	MSB	0	0	1	1	1								
CH-2 DMA Address	LSB	0	1	0	0	0	Same as Channel 0							
	MSB	0	1	0	0	1								
CH-2 Terminal Count	LSB	0	1	0	1	0	Same as Channel 0							
	MSB	0	1	0	1	1								
CH-3 DMA Address	LSB	0	1	1	0	0	Same as Channel 0							
	MSB	0	1	1	0	1								
CH-3 Terminal Count	LSB	0	1	1	1	0								
	MSB	0	1	1	1	1								
MODE SET (Program only)	—	1	0	0	0	0	AL	TCS	EW	RP	EN3	EN2	EN1	EN0
STATUS (Read only)	—	1	0	0	0	0	0	0	0	UP	TC3	TC2	TC1	TC0

\* $A_0$ - $A_{15}$ : DMA Starting Address,  $C_0$ - $C_{13}$ : Terminal Count value (N-1), Rd and Wr: DMA Verify (00), Write (01) or Read (10) cycle selection, AL: Auto Load, TCS: TC STOP, EW: EXTENDED WRITE, RP: ROTATING PRIORITY, EN3-EN0: CHANNEL ENABLE MASK, UP: UPDATE FLAG, TC3-TC0: TERMINAL COUNT STATUS BITS.





1 DRQn refers to any DRQ line on an enabled DMA channel.

Figure 6. DMA Operation State Diagram

## DMA OPERATION

### Single Byte Transfers

A single byte transfer is initiated by the I/O device raising the DRQ line of one channel of the 8257. If the channel is enabled, the 8257 will output a HRQ to the CPU. The 8257 now waits until a HLDA is received insuring that the system bus is free for its use. Once HLDA is received the  $\overline{\text{DACK}}$  line for the requesting channel is activated (LOW). The  $\overline{\text{DACK}}$  line acts as a chip select for the requesting I/O device. The 8257 then generates the

read and write commands and byte transfer occurs between the selected I/O device and memory. After the transfer is complete, the  $\overline{\text{DACK}}$  line is set HIGH and the HRQ line is set LOW to indicate to the CPU that the bus is now free for use. DRQ must remain HIGH until  $\overline{\text{DACK}}$  is issued to be recognized and must go LOW before S4 of the transfer sequence to prevent another transfer from occurring. (See timing diagram.)

### Consecutive Transfers

If more than one channel requests service simultaneously, the transfer will occur in the same way a burst does. No overhead is incurred by switching from one channel to another. In each S4 the DRQ lines are sampled and the highest priority request is recognized during the next transfer. A burst mode transfer in a lower priority channel will be overridden by a higher priority request. Once the high priority transfer has completed control will return to the lower priority channel if its DRQ is still active. No extra cycles are needed to execute this sequence and the HRQ line remains active until all DRQ lines go LOW.

### Control Override

The continuous DMA transfer mode described above can be interrupted by an external device by lowering the HLDA line. After each DMA transfer the 8257 samples the HLDA line to insure that it is still active. If it is not active, the 8257 completes the current transfer, releases the HRQ line (LOW) and returns to the idle state. If DRQ lines are still active the 8257 will raise the HRQ line in the third cycle and proceed normally. (See timing diagram.)

### Not Ready

The 8257 has a Ready input similar to the 8080A and the 8085A. The Ready line is sampled in State 3. If Ready is LOW the 8257 enters a wait state. Ready is sampled during every wait state. When Ready returns HIGH the 8257 proceeds to State 4 to complete the transfer. Ready is used to interface memory or I/O devices that cannot meet the bus set up times required by the 8257.

### Speed

The 8257 uses four clock cycles to transfer a byte of data. No cycles are lost in the master to master transfer maximizing bus efficiency. A 2MHz clock input will allow the 8257 to transfer at a rate of 500K bytes/second.

### Memory Mapped I/O Configurations

The 8257 can be connected to the system bus as a memory device instead of as an I/O device for memory mapped I/O configurations by connecting the system memory control lines to the 8257's I/O control lines and the system I/O control lines to the 8257's memory control lines.

This configuration permits use of the 8080's considerably larger repertoire of memory instructions when reading or loading the 8257's registers. Note that with this connection, the programming of the Read (bit 15) and Write (bit 14) bits in the terminal count register will have a different meaning:



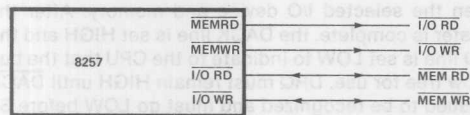


Figure 7. System Interface for Memory Mapped I/O

BIT 15 READ	BIT 14 WRITE	
0	0	DMA Verify Cycle
0	1	DMA Read Cycle
1	0	DMA Write Cycle
1	1	Illegal

Figure 8. TC Register for Memory Mapped I/O Only

## SYSTEM APPLICATION EXAMPLES

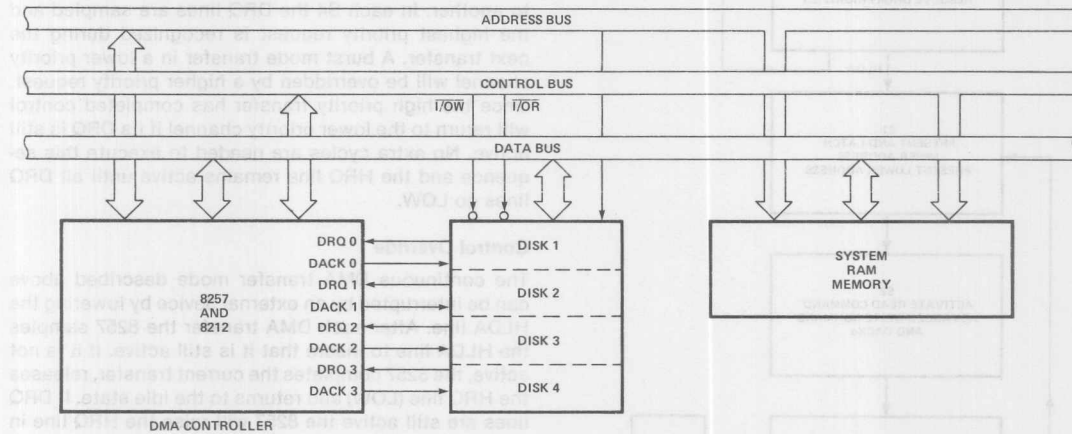


Figure 9. Floppy Disk Controller (4 Drives)

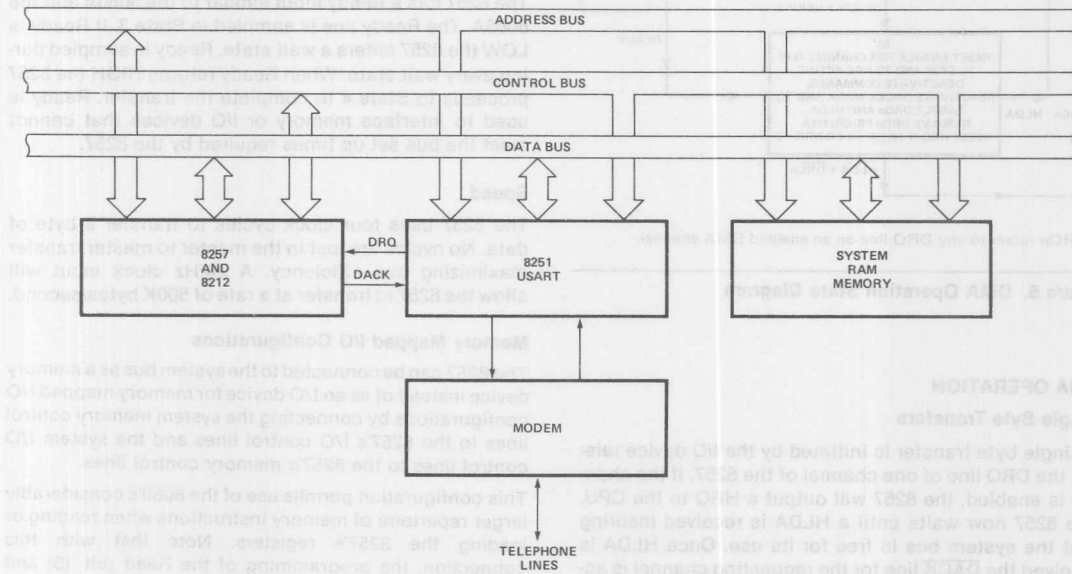


Figure 10. High-Speed Communication Controller

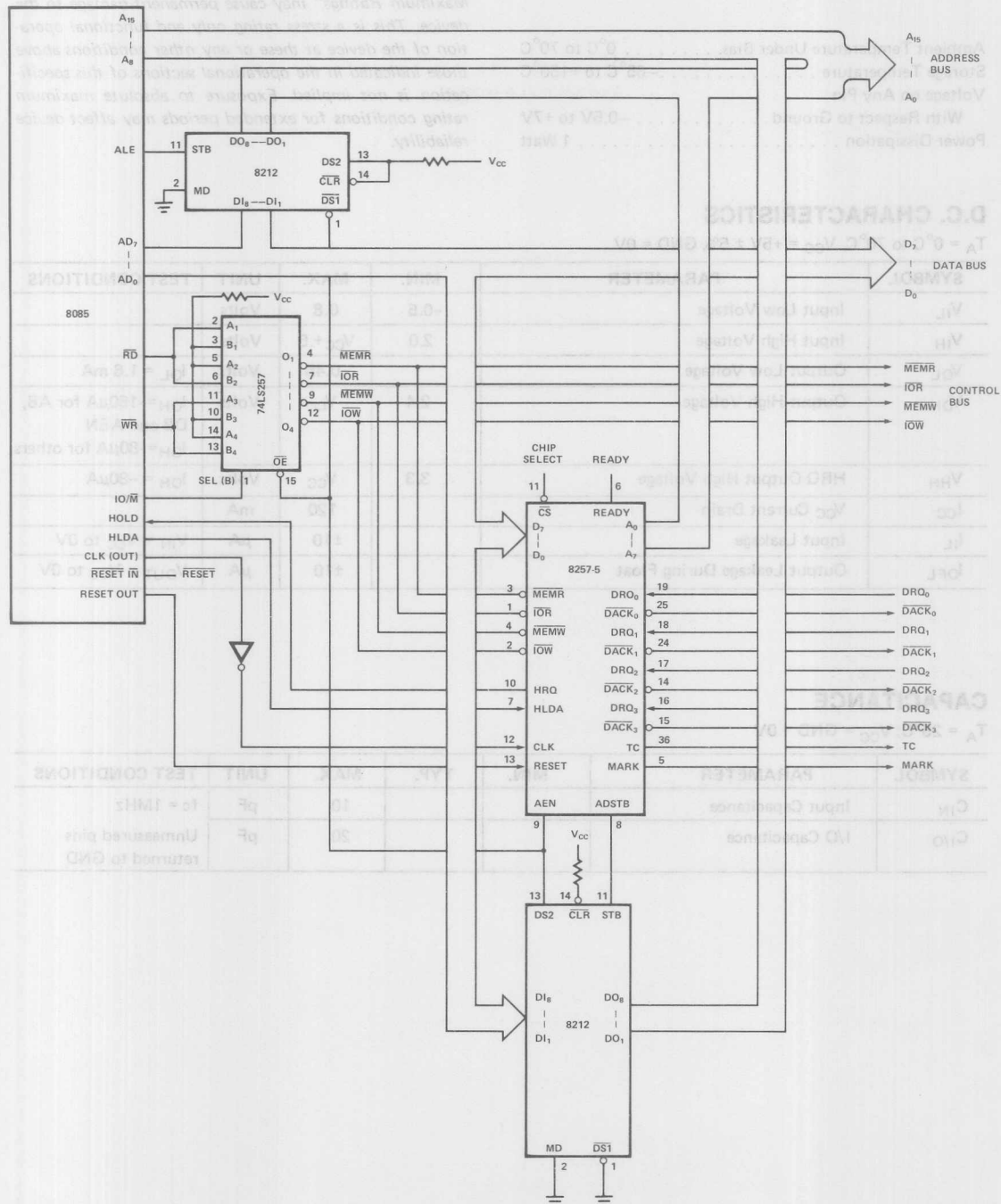


Figure 11. Detailed System Interface Schematic

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias. . . . . 0°C to 70°C  
 Storage Temperature . . . . . -65°C to +150°C  
 Voltage on Any Pin  
     With Respect to Ground. . . . . -0.5V to +7V  
 Power Dissipation . . . . . 1 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS**

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = +5V \pm 5\%$ , GND = 0V

SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
$V_{IL}$	Input Low Voltage	-0.5	0.8	Volts	
$V_{IH}$	Input High Voltage	2.0	$V_{CC} + 5$	Volts	
$V_{OL}$	Output Low Voltage		0.45	Volts	$I_{OL} = 1.6\text{ mA}$
$V_{OH}$	Output High Voltage	2.4	$V_{CC}$	Volts	$I_{OH} = -150\mu\text{A}$ for AB, DB and AEN $I_{OH} = -80\mu\text{A}$ for others
$V_{HH}$	HRQ Output High Voltage	3.3	$V_{CC}$	Volts	$I_{OH} = -80\mu\text{A}$
$I_{CC}$	$V_{CC}$ Current Drain		120	mA	
$I_{IL}$	Input Leakage		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ to 0V
$I_{OFL}$	Output Leakage During Float		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC}$ to 0V

**CAPACITANCE**

$T_A = 25^\circ\text{C}$ ;  $V_{CC} = \text{GND} = 0V$

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
$C_{IN}$	Input Capacitance			10	pF	$f_c = 1\text{ MHz}$
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to GND

**A.C. CHARACTERISTICS: PERIPHERAL (SLAVE) MODE**

$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 5\%$ ;  $\text{GND} = 0\text{V}$  (Note 1).

**8080 Bus Parameters****Read Cycle:**

Symbol	Parameter	8257		8257-5		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
$T_{AR}$	Adr or $\overline{\text{CS}}\downarrow$ Setup to $\overline{\text{RD}}\downarrow$	0		0		ns	
$T_{RA}$	Adr or $\overline{\text{CS}}\uparrow$ Hold from $\overline{\text{RD}}\uparrow$	0		0		ns	
$T_{RD}$	Data Access from $\overline{\text{RD}}\downarrow$	0	300	0	200	ns	(Note 2)
$T_{DF}$	DB $\rightarrow$ Float Delay from $\overline{\text{RD}}\uparrow$	20	150	20	100	ns	
$T_{RR}$	$\overline{\text{RD}}$ Width	250		250		ns	

**Write Cycle:**

Symbol	Parameter	8257		8257-5		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
$T_{AW}$	Adr Setup to $\overline{\text{WR}}\downarrow$	20		20		ns	
$T_{WA}$	Adr Hold from $\overline{\text{WR}}\uparrow$	0		0		ns	
$T_{DW}$	Data Setup to $\overline{\text{WR}}\uparrow$	200		200		ns	
$T_{WD}$	Data Hold from $\overline{\text{WR}}\uparrow$	0		0		ns	
$T_{WW}$	$\overline{\text{WR}}$ Width	200		200		ns	

**Other Timing:**

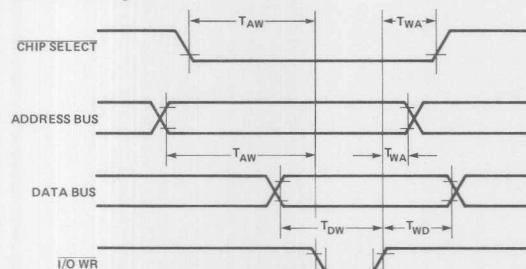
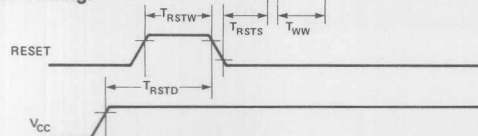
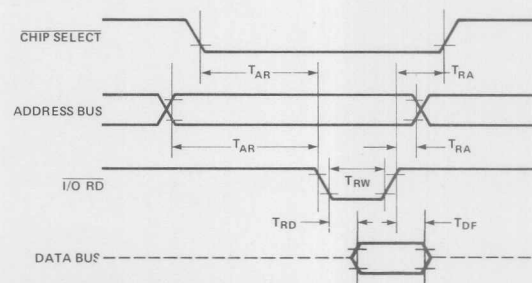
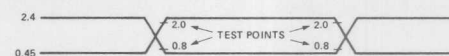
Symbol	Parameter	8257		8257-5		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
$T_{RSTW}$	Reset Pulse Width	300		300		ns	
$T_{RSTD}$	Power Supply $\uparrow$ ( $V_{CC}$ ) Setup to Reset $\downarrow$	500		500		$\mu\text{s}$	
$T_r$	Signal Rise Time		20		20	ns	
$T_f$	Signal Fall Time		20		20	ns	
$T_{RSTS}$	Reset to First $\overline{\text{I/O}}\overline{\text{WR}}$	2		2		$t_{CY}$	

Notes: 1. All timing measurements are made at the following reference voltages unless specified otherwise:

Input "1" at 2.0V, "0" at 0.8V

2. 8257:  $C_L = 100\text{pF}$ , 8257-5:  $C_L = 150\text{pF}$ .

Output "1" at 2.0V, "0" at 0.8V

**8257 PERIPHERAL MODE TIMING DIAGRAMS****Write Timing:****Reset Timing:****Read Timing:****Input Waveform for A.C. Tests:**

## Timing Requirements

SYMBOL	PARAMETER	8257		8257-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
$T_{CY}$	Cycle Time (Period)	0.320	4	0.320	4	$\mu s$
$T_{\theta}$	Clock Active (High)	120	$.8T_{CY}$	80	$.8T_{CY}$	ns
$T_{QS}$	DRQ $\uparrow$ Setup to $\theta\downarrow$ (SI, S4)	120		120		ns
$T_{QH}$	DRQ $\downarrow$ Hold from HLDA $\uparrow$ [4]	0		0		ns
$T_{HS}$	HLDA $\uparrow$ or $\downarrow$ Setup to $\theta\downarrow$ (SI, S4)	100		100		ns
$T_{RS}$	READY Setup Time to $\theta\uparrow$ (S3, Sw)	30		30		ns
$T_{RH}$	READY Hold Time from $\theta\uparrow$ (S3, Sw)	20		20		ns

Note: 4. Tracking Parameter.

## Tracking Parameters

Signals labeled as Tracking Parameters (footnotes 4-7 under A.C. Specifications) are signals that follow similar paths through the silicon die. The propagation speed of these signals varies in the manufacturing process but the relationship between all these parameters is constant. The variation is less than or equal to 50 ns.

Suppose the following timing equation is being evaluated,

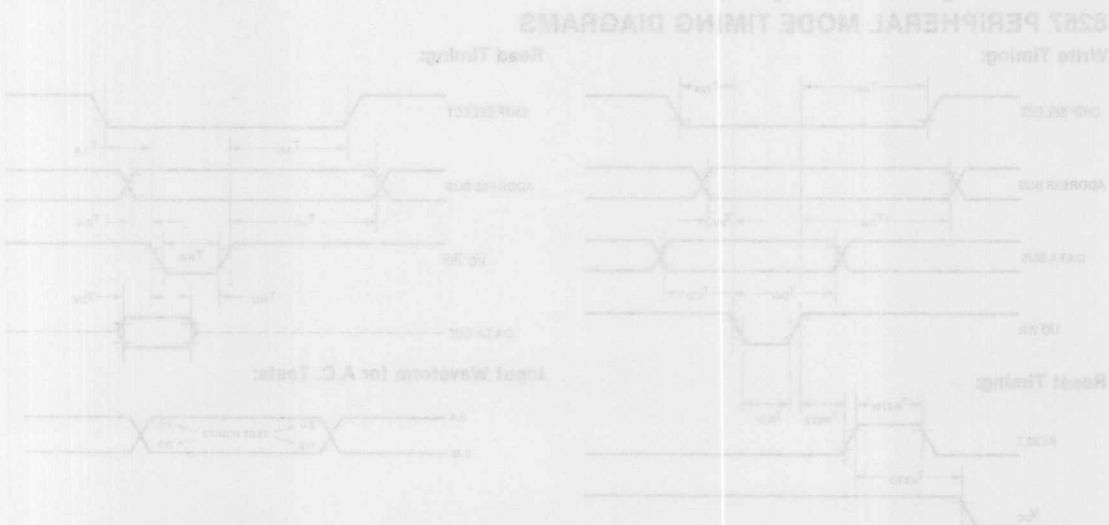
$$T_{A(MIN)} + T_{B(MAX)} \leq 150 \text{ ns}$$

and only minimum specifications exist for  $T_A$  and  $T_B$ . If  $T_{A(MIN)}$  is used, and if  $T_A$  and  $T_B$  are tracking parameters,  $T_{B(MAX)}$  can be taken as  $T_{B(MIN)} + 50 \text{ ns}$ .

$$T_{A(MIN)} + (T_{B(MIN)} + 50 \text{ ns}) \leq 150 \text{ ns}$$

\*if  $T_A$  and  $T_B$  are tracking parameters

Figure 1. All timing measurements are made at the following reference voltage levels unless otherwise specified: Input "1" at 2.0V, "0" at 0.5V. Output "1" at 2.0V, "0" at 0.5V.





**A.C. CHARACTERISTICS: DMA (MASTER) MODE**  $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = +5\text{V} \pm 5\%$ ,  $\text{GND} = 0\text{V}$ **Timing Responses**

SYMBOL	PARAMETER	8257		8257-5		UNIT
		MIN.	MAX.	MIN.	MAX.	
$T_{DQ}$	HRQ $\uparrow$ or $\downarrow$ Delay from $\theta\uparrow$ (S1,S4) (measured at 2.0V) <sup>[1]</sup>		160		160	ns
$T_{DQ1}$	HRQ $\uparrow$ or $\downarrow$ Delay from $\theta\uparrow$ (S1,S4) (measured at 3.3V) <sup>[3]</sup>		250		250	ns
$T_{AEL}$	AEN $\uparrow$ Delay from $\theta\downarrow$ (S1) <sup>[1]</sup>		300		300	ns
$T_{AET}$	AEN $\downarrow$ Delay from $\theta\uparrow$ (S1) <sup>[1]</sup>		200		200	ns
$T_{AEA}$	Adr(AB)(Active) Delay from AEN $\uparrow$ (S1) <sup>[4]</sup>	20		20		ns
$T_{FAAB}$	Adr(AB)(Active) Delay from $\theta\uparrow$ (S1) <sup>[2]</sup>		250		250	ns
$T_{AFAB}$	Adr(AB)(Float) Delay from $\theta\uparrow$ (S1) <sup>[2]</sup>		150		150	ns
$T_{ASM}$	Adr(AB)(Stable) Delay from $\theta\uparrow$ (S1) <sup>[2]</sup>		250		250	ns
$T_{AH}$	Adr(AB)(Stable) Hold from $\theta\uparrow$ (S1) <sup>[2]</sup>	$T_{ASM}-50$		$T_{ASM}-50$		ns
$T_{AHR}$	Adr(AB)(Valid) Hold from $\overline{\text{Rd}}\uparrow$ (S1,S1) <sup>[4]</sup>	60		60		ns
$T_{AHW}$	Adr(AB)(Valid) Hold from $\overline{\text{Wr}}\uparrow$ (S1,S1) <sup>[4]</sup>	300		300		ns
$T_{FADB}$	Adr(DB)(Active) Delay from $\theta\uparrow$ (S1) <sup>[2]</sup>		300		300	ns
$T_{AFDB}$	Adr(DB)(Float) Delay from $\theta\uparrow$ (S2) <sup>[2]</sup>	$T_{STT}+20$	250	$T_{STT}+20$	170	ns
$T_{ASS}$	Adr(DB) Setup to AdrStb $\downarrow$ (S1-S2) <sup>[4]</sup>	100		100		ns
$T_{AHS}$	Adr(DB)(Valid) Hold from AdrStb $\downarrow$ (S2) <sup>[4]</sup>	50		50		ns
$T_{STL}$	AdrStb $\uparrow$ Delay from $\theta\uparrow$ (S1) <sup>[1]</sup>		200		200	ns
$T_{STT}$	AdrStb $\downarrow$ Delay from $\theta\uparrow$ (S2) <sup>[1]</sup>		140		140	ns
$T_{SW}$	AdrStb Width (S1-S2) <sup>[4]</sup>	$T_{CY}-100$		$T_{CY}-100$		ns
$T_{ASC}$	$\overline{\text{Rd}}\downarrow$ or $\overline{\text{Wr}}(\text{Ext})\downarrow$ Delay from AdrStb $\downarrow$ (S2) <sup>[4]</sup>	70		70		ns
$T_{DBC}$	$\overline{\text{Rd}}\downarrow$ or $\overline{\text{Wr}}(\text{Ext})\downarrow$ Delay from Adr(DB) (Float)(S2) <sup>[4]</sup>	20		20		ns
$T_{AK}$	DACK $\uparrow$ or $\downarrow$ Delay from $\theta\downarrow$ (S2,S1) and TC/Mark $\uparrow$ Delay from $\theta\uparrow$ (S3) and TC/Mark $\downarrow$ Delay from $\theta\uparrow$ (S4) <sup>[1,5]</sup>		250		250	ns
$T_{DCL}$	$\overline{\text{Rd}}\downarrow$ or $\overline{\text{Wr}}(\text{Ext})\downarrow$ Delay from $\theta\uparrow$ (S2) and $\overline{\text{Wr}}\downarrow$ Delay from $\theta\uparrow$ (S3) <sup>[2,6]</sup>		200		200	ns
$T_{DCT}$	$\overline{\text{Rd}}\uparrow$ Delay from $\theta\downarrow$ (S1,S1) and $\overline{\text{Wr}}\uparrow$ Delay from $\theta\uparrow$ (S4) <sup>[2,7]</sup>		200		200	ns
$T_{FAC}$	$\overline{\text{Rd}}$ or $\overline{\text{Wr}}$ (Active) from $\theta\uparrow$ (S1) <sup>[2]</sup>		300		300	ns
$T_{AFC}$	$\overline{\text{Rd}}$ or $\overline{\text{Wr}}$ (Float) from $\theta\uparrow$ (S1) <sup>[2]</sup>		150		150	ns
$T_{RWM}$	$\overline{\text{Rd}}$ Width (S2-S1 or S1) <sup>[4]</sup>	$2T_{CY} + T_\theta - 50$		$2T_{CY} + T_\theta - 50$		ns
$T_{WWM}$	$\overline{\text{Wr}}$ Width (S3-S4) <sup>[4]</sup>	$T_{CY}-50$		$T_{CY}-50$		ns
$T_{WWME}$	$\overline{\text{Wr}}(\text{Ext})$ Width (S2-S4) <sup>[4]</sup>	$2T_{CY}-50$		$2T_{CY}-50$		ns

Notes: 1. Load = 1 TTL. 2. Load = 1 TTL + 50pF. 3. Load = 1 TTL + ( $R_L = 3.3\text{K}$ ),  $V_{OH} = 3.3\text{V}$ . 4. Tracking Parameter.  
5.  $\Delta T_{AK} < 50\text{ ns}$ . 6.  $\Delta T_{DCL} < 50\text{ ns}$ . 7.  $\Delta T_{DCT} < 50\text{ ns}$ .

[illegible]

**NOTE:** The clock waveform is duplicated for clarity. The 8257 requires only one clock input.

**Figure 12. Consecutive Cycles and Burst Mode Sequence**

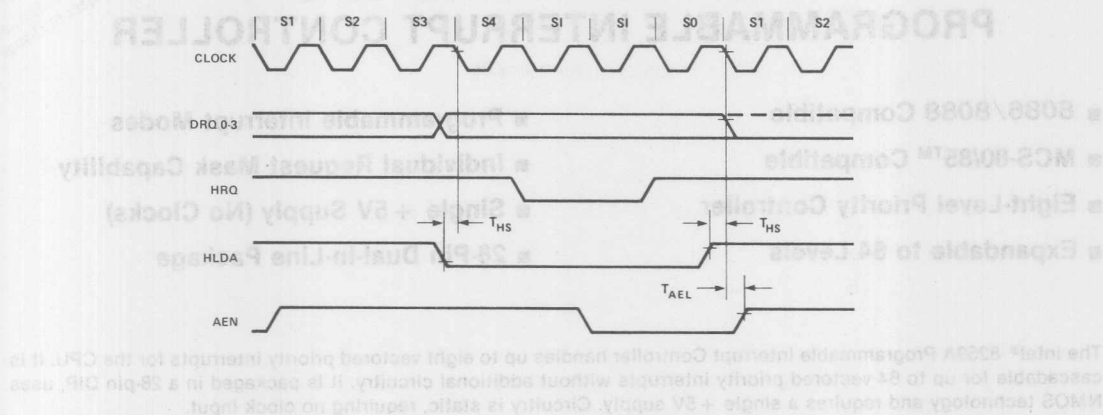


Figure 13. Control Override Sequence

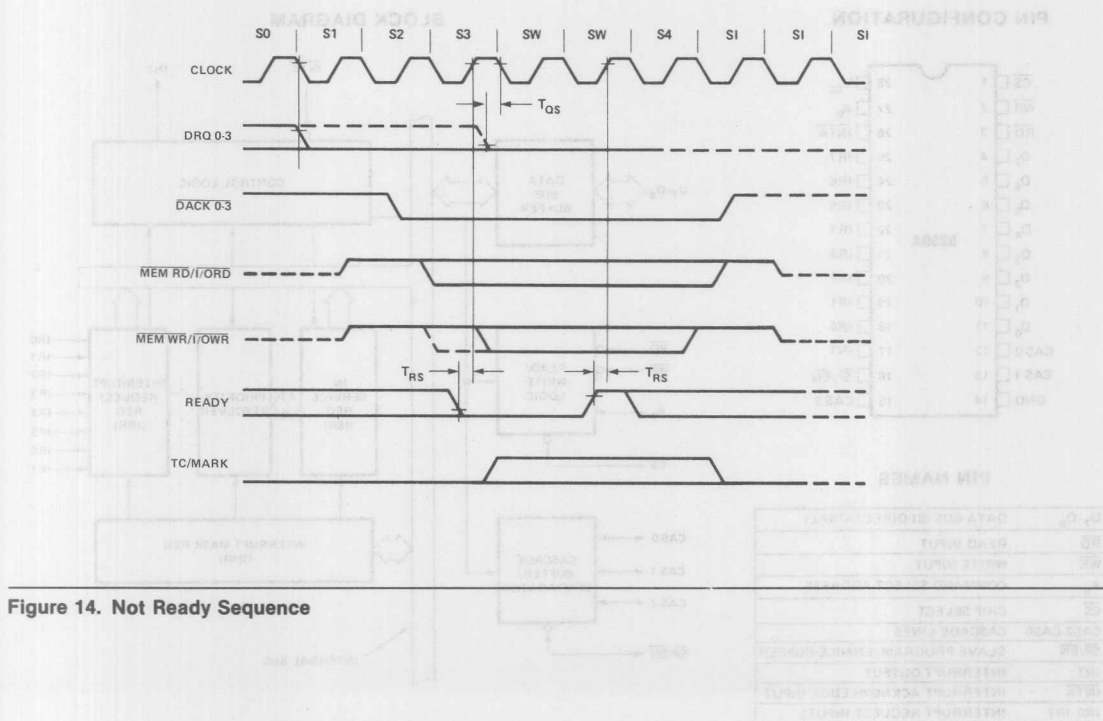


Figure 14. Not Ready Sequence

## 8259A/8259A-2/8259A-8 PROGRAMMABLE INTERRUPT CONTROLLER

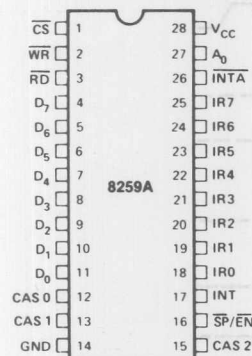
- 8086/8088 Compatible
- MCS-80/85™ Compatible
- Eight-Level Priority Controller
- Expandable to 64 Levels
- Programmable Interrupt Modes
- Individual Request Mask Capability
- Single +5V Supply (No Clocks)
- 28-Pin Dual-In-Line Package

The Intel® 8259A Programmable Interrupt Controller handles up to eight vectored priority interrupts for the CPU. It is cascadable for up to 64 vectored priority interrupts without additional circuitry. It is packaged in a 28-pin DIP, uses NMOS technology and requires a single +5V supply. Circuitry is static, requiring no clock input.

The 8259A is designed to minimize the software and real time overhead in handling multi-level priority interrupts. It has several modes, permitting optimization for a variety of system requirements.

The 8259A is fully upward compatible with the Intel® 8259. Software originally written for the 8259 will operate the 8259A in all 8259 equivalent modes (MCS-80/85, Non-Buffered, Edge Triggered).

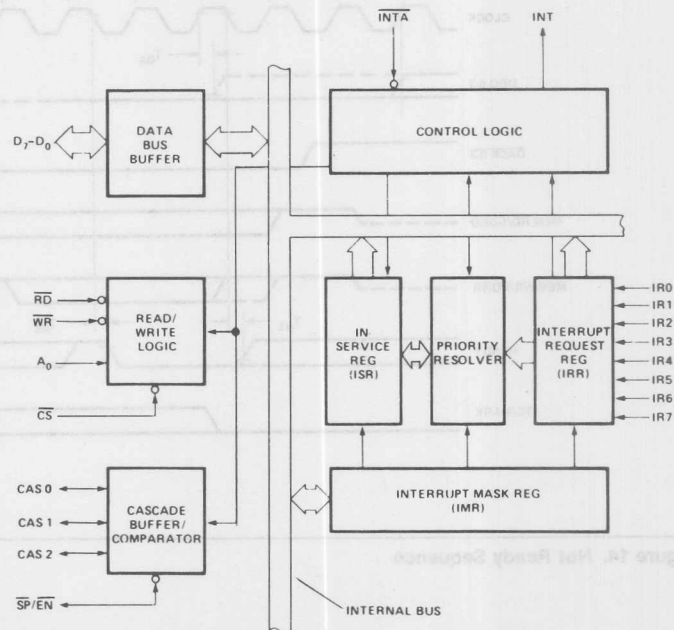
### PIN CONFIGURATION



### PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	DATA BUS (BI-DIRECTIONAL)
RD	READ INPUT
WR	WRITE INPUT
A <sub>0</sub>	COMMAND SELECT ADDRESS
CS	CHIP SELECT
CAS2 CAS0	CASCADE LINES
SP/EN	SLAVE PROGRAM / ENABLE BUFFER
INT	INTERRUPT OUTPUT
INTA	INTERRUPT ACKNOWLEDGE INPUT
IR0-IR7	INTERRUPT REQUEST INPUTS

### BLOCK DIAGRAM



## INTERRUPTS IN MICROCOMPUTER SYSTEMS

Microcomputer system design requires that I/O devices such as keyboards, displays, sensors and other components receive servicing in an efficient manner so that large amounts of the total system tasks can be assumed by the microcomputer with little or no effect on throughput.

The most common method of servicing such devices is the *Polled* approach. This is where the processor must test each device in sequence and in effect "ask" each one if it needs servicing. It is easy to see that a large portion of the main program is looping through this continuous polling cycle and that such a method would have a serious, detrimental effect on system throughput, thus limiting the tasks that could be assumed by the microcomputer and reducing the cost effectiveness of using such devices.

A more desirable method would be one that would allow the microprocessor to be executing its main program and only stop to service peripheral devices when it is told to do so by the device itself. In effect, the method would provide an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executed and fetch a new routine that will service the requesting device. Once this servicing is complete, however, the processor would resume exactly where it left off.

This method is called *Interrupt*. It is easy to see that system throughput would drastically increase, and thus more tasks could be assumed by the microcomputer to further enhance its cost effectiveness.

The Programmable Interrupt Controller (PIC) functions as an overall manager in an Interrupt-Driven system environment. It accepts requests from the peripheral equipment, determines which of the incoming requests is of the highest importance (priority), ascertains whether the incoming request has a higher priority value than the level currently being serviced, and issues an interrupt to the CPU based on this determination.

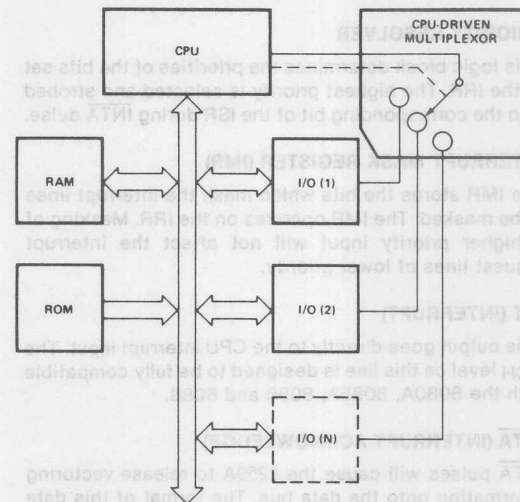
Each peripheral device or structure usually has a special program or "routine" that is associated with its specific functional or operational requirements; this is referred to as a "service routine". The PIC, after issuing an interrupt to the CPU, must somehow input information into the CPU that can "point" the Program Counter to the service routine associated with the requesting device. This "pointer" is an address in a vectoring table and will often be referred to, in this document, as vectoring data.

## 8259A BASIC FUNCTIONAL DESCRIPTION

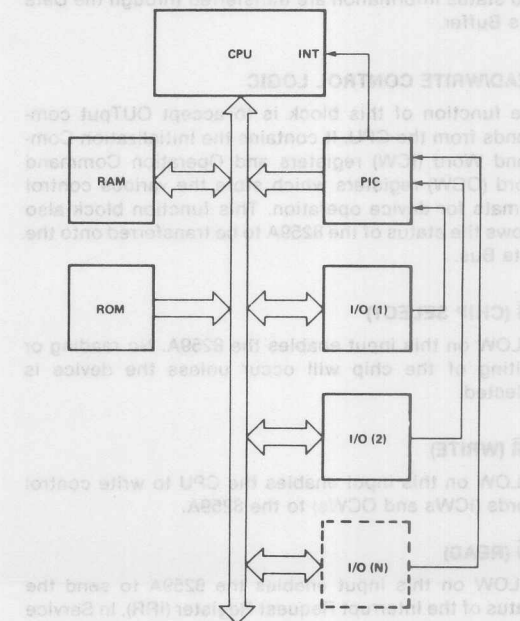
### GENERAL

The 8259A is a device specifically designed for use in real time, interrupt driven microcomputer systems. It manages eight levels or requests and has built-in features for expandability to other 8259A's (up to 64 levels). It is programmed by the system's software as an I/O peripheral. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by the 8259A can be configured to

match his system requirements. The priority modes can be changed or reconfigured dynamically at any time during the main program. This means that the complete interrupt structure can be defined as required based on the total system environment.



Polled Method



Interrupt Method



### INTERRUPT REQUEST REGISTER (IRR) AND IN-SERVICE REGISTER (ISR)

The interrupts at the IR input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service Register (ISR). The IRR is used to store all the interrupt levels which are requesting service; and the ISR is used to store all the interrupt levels which are being serviced.

### PRIORITY RESOLVER

This logic block determines the priorities of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during INTA pulse.

### INTERRUPT MASK REGISTER (IMR)

The IMR stores the bits which mask the interrupt lines to be masked. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority.

### INT (INTERRUPT)

This output goes directly to the CPU interrupt input. The  $V_{OH}$  level on this line is designed to be fully compatible with the 8080A, 8085A, 8086 and 8088.

### INTA (INTERRUPT ACKNOWLEDGE)

INTA pulses will cause the 8259A to release vectored information onto the data bus. The format of this data depends on the system mode ( $\mu PM$ ) of the 8259A.

### DATA BUS BUFFER

This 3-state, bidirectional 8-bit buffer is used to interface the 8259A to the system Data Bus. Control words and status information are transferred through the Data Bus Buffer.

### READ/WRITE CONTROL LOGIC

The function of this block is to accept OUTput commands from the CPU. It contains the Initialization Command Word (ICW) registers and Operation Command Word (OCW) registers which store the various control formats for device operation. This function block also allows the status of the 8259A to be transferred onto the Data Bus.

### $\overline{CS}$ (CHIP SELECT)

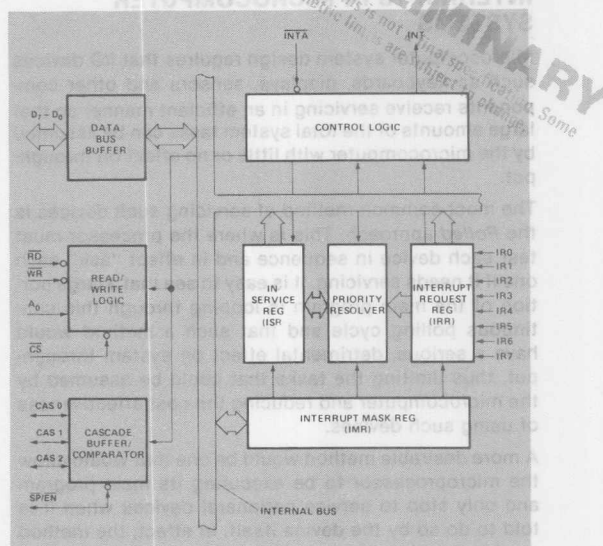
A LOW on this input enables the 8259A. No reading or writing of the chip will occur unless the device is selected.

### $\overline{WR}$ (WRITE)

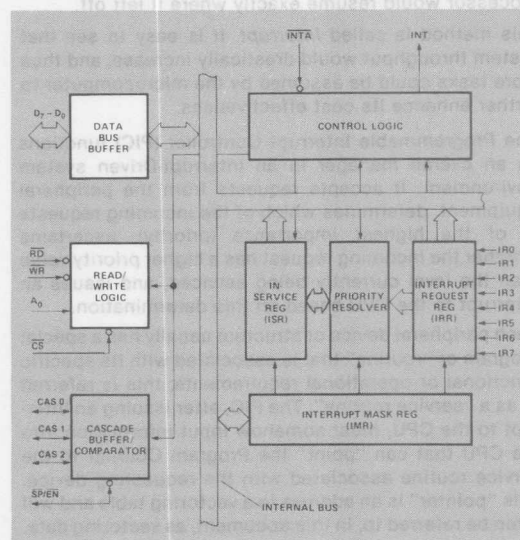
A LOW on this input enables the CPU to write control words (ICWs and OCWs) to the 8259A.

### $\overline{RD}$ (READ)

A LOW on this input enables the 8259A to send the status of the Interrupt Request Register (IRR), In Service Register (ISR), the Interrupt Mask Register (IMR), or the Interrupt level onto the Data Bus.



8259A Block Diagram



8259A Block Diagram

### $A_0$

This input signal is used in conjunction with  $\overline{WR}$  and  $\overline{RD}$  signals to write commands into the various command registers, as well as reading the various status registers of the chip. This line can be tied directly to one of the address lines.

### THE CASCADE BUFFER/COMPARATOR

This function block stores and compares the IDs of all 8259A's used in the system. The associated three I/O pins (CAS0-2) are outputs when the 8259A is used as a master and are inputs when the 8259A is used as a slave. As a master, the 8259A sends the ID of the interrupting slave device onto the CAS0-2 lines. The slave thus selected will send its preprogrammed subroutine address onto the Data Bus during the next one or two consecutive  $\overline{INTA}$  pulses. (See section "Cascading the 8259A".)

### INTERRUPT SEQUENCE

The powerful features of the 8259A in a microcomputer system are its programmability and the interrupt routine addressing capability. The latter allows direct or indirect jumping to the specific interrupt routine requested without any polling of the interrupting devices. The normal sequence of events during an interrupt depends on the type of CPU being used.

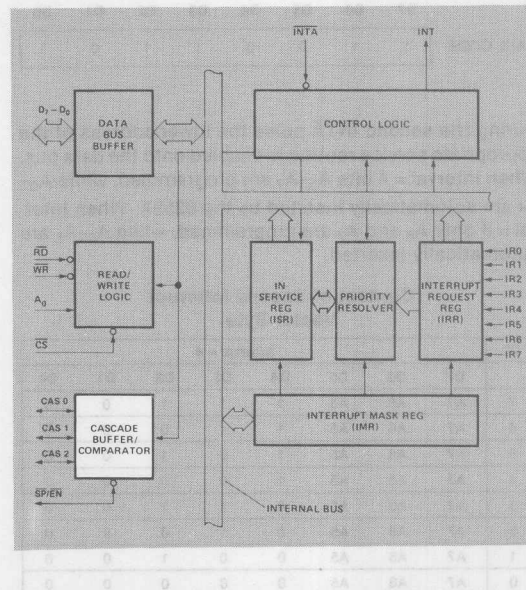
The events occur as follows in an MCS-80/85 system:

1. One or more of the INTERRUPT REQUEST lines (IR7-0) are raised high, setting the corresponding IRR bit(s).
2. The 8259A evaluates these requests, and sends an  $\overline{INT}$  to the CPU, if appropriate.
3. The CPU acknowledges the  $\overline{INT}$  and responds with an  $\overline{INTA}$  pulse.
4. Upon receiving an  $\overline{INTA}$  from the CPU group, the highest priority ISR bit is set, and the corresponding IRR bit is reset. The 8259A will also release a CALL instruction code (11001101) onto the 8-bit Data Bus through its D7-0 pins.
5. This CALL instruction will initiate two more  $\overline{INTA}$  pulses to be sent to the 8259A from the CPU group.
6. These two  $\overline{INTA}$  pulses allow the 8259A to release its preprogrammed subroutine address onto the Data Bus. The lower 8-bit address is released at the first  $\overline{INTA}$  pulse and the higher 8-bit address is released at the second  $\overline{INTA}$  pulse.
7. This completes the 3-byte CALL instruction released by the 8259A. In the AEOI mode the ISR bit is reset at the end of the third  $\overline{INTA}$  pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt sequence.

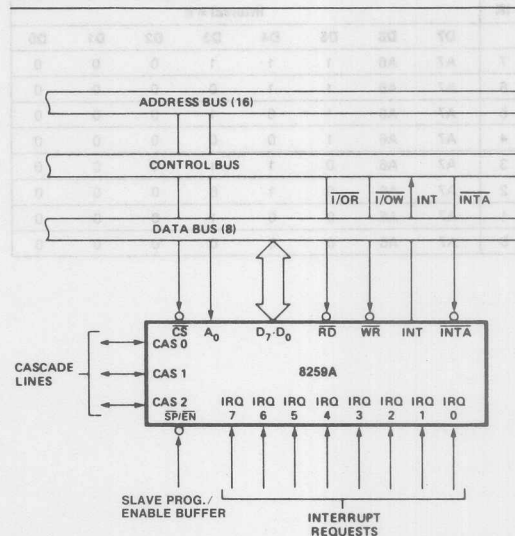
The events occurring in an 8086/8088 system are the same until step 4.

4. Upon receiving an  $\overline{INTA}$  from the CPU group, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259A does not drive the Data Bus during this cycle.
5. The 8086/8088 CPU will initiate a second  $\overline{INTA}$  pulse. During this pulse, the 8259A releases an 8-bit pointer onto the Data Bus where it is read by the CPU.
6. This completes the interrupt cycle. In the AEOI mode the ISR bit is reset at the end of the second  $\overline{INTA}$  pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

If no interrupt request is present at step 4 of either sequence (i.e., the request was too short in duration) the 8259A will issue an interrupt level 7. Both the vectored bytes and the CAS lines will look like an interrupt level 7 was requested.



8259A Block Diagram



8259A Interface to Standard System Bus

## INTERRUPT SEQUENCE OUTPUTS

### MCS-80/85 MODE

This sequence is timed by three INTA pulses. During the first INTA pulse the CALL opcode is enabled onto the data bus.

Content of First Interrupt Vector Byte

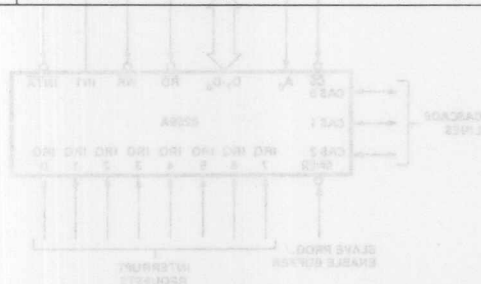
	D7	D6	D5	D4	D3	D2	D1	D0
CALL CODE	1	1	0	0	1	1	0	1

During the second INTA pulse the lower address of the appropriate service routine is enabled onto the data bus. When Interval = 4 bits A<sub>5</sub>-A<sub>7</sub> are programmed, while A<sub>0</sub>-A<sub>4</sub> are automatically inserted by the 8259A. When Interval = 8 only A<sub>6</sub> and A<sub>7</sub> are programmed, while A<sub>0</sub>-A<sub>5</sub> are automatically inserted.

Content of Second Interrupt Vector Byte

IR	Interval = 4							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	A5	1	1	1	0	0
6	A7	A6	A5	1	1	0	0	0
5	A7	A6	A5	1	0	1	0	0
4	A7	A6	A5	1	0	0	0	0
3	A7	A6	A5	0	1	1	0	0
2	A7	A6	A5	0	1	0	0	0
1	A7	A6	A5	0	0	1	0	0
0	A7	A6	A5	0	0	0	0	0

IR	Interval = 8							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	1	1	1	0	0	0
6	A7	A6	1	1	0	0	0	0
5	A7	A6	1	0	1	0	0	0
4	A7	A6	1	0	0	0	0	0
3	A7	A6	0	1	1	0	0	0
2	A7	A6	0	1	0	0	0	0
1	A7	A6	0	0	1	0	0	0
0	A7	A6	0	0	0	0	0	0



During the third INTA pulse the higher address of the appropriate service routine, which was programmed as byte 2 of the initialization sequence (A<sub>8</sub>-A<sub>15</sub>), is enabled onto the bus.

Content of Third Interrupt Vector Byte

D7	D6	D5	D4	D3	D2	D1	D0
A15	A14	A13	A12	A11	A10	A9	A8

### 8086/8088 Mode

8086/8088 mode is similar to MCS80/85 mode except that only two Interrupt Acknowledge cycles are issued by the processor and no CALL opcode is sent to the processor. The first interrupt acknowledge cycle is similar to that of MCS-80/85 systems in that the 8259A uses it to internally freeze the state of the interrupts for priority resolution and as a master it issues the interrupt code on the cascade lines at the end of the INTA pulse. On this first cycle it does not issue any data to the processor and leaves its data bus buffers disabled. On the second interrupt acknowledge cycle in 8086/8088 mode the master (or slave if so programmed) will send a byte of data to the processor with the acknowledged interrupt code composed as follows (note the state of the ADI mode control is ignored and A<sub>5</sub>-A<sub>11</sub> are unused in 8086/8088 mode):

	D7	D6	D5	D4	D3	D2	D1	D0
IR7	T7	T6	T5	T4	T3	1	1	1
IR6	T7	T6	T5	T4	T3	1	1	0
IR5	T7	T6	T5	T4	T3	1	0	1
IR4	T7	T6	T5	T4	T3	1	0	0
IR3	T7	T6	T5	T4	T3	0	1	1
IR2	T7	T6	T5	T4	T3	0	1	0
IR1	T7	T6	T5	T4	T3	0	0	1
IR0	T7	T6	T5	T4	T3	0	0	0

## PROGRAMMING THE 8259A

The 8259A accepts two types of command words generated by the CPU:

1. **Initialization Command Words (ICWs):** Before normal operation can begin, each 8259A in the system must be brought to a starting point — by a sequence of 2 to 4 bytes timed by WR pulses. This sequence is described in Figure 1.
2. **Operation Command Words (OCWs):** These are the command words that are sent to the 8259A for various forms of operation, such as:
  - Interrupt Masking
  - End of Interrupt
  - Priority Rotation
  - Interrupt Status

The OCWs can be written into the 8259A anytime after initialization.

## INITIALIZATION

## GENERAL

Whenever a command is issued with A0=0 and D4=1, this is interpreted as Initialization Command Word 1 (ICW1). ICW1 starts the initialization sequence during which the following automatically occur.

- a. The edge sense circuit is reset, which means that following initialization, an interrupt request (IR) input must make a low-to-high transition to generate an interrupt.
- b. The Interrupt Mask Register is cleared.
- c. R7 input is assigned priority 7.
- d. The slave mode address is set to 7.
- e. Special Mask Mode is cleared and Status Read is set to IRR.
- f. If IC4=0, then all functions selected in ICW4 are set to zero. (Non-Buffered mode\*, no Auto-EOI, MCS-80/85 system).

\*Note: Master/Slave in ICW4 is only used in the buffered mode.

A <sub>0</sub>	D <sub>4</sub>	D <sub>3</sub>	RD	WR	CS	INPUT OPERATION (READ)
0			0	1	0	IRR, ISR or Interrupting Level → DATA BUS (Note 1)
1			0	1	0	IMR → DATA BUS
						OUTPUT OPERATION (WRITE)
0	0	0	1	0	0	DATA BUS → OCW2
0	0	1	1	0	0	DATA BUS → OCW3
0	1	X	1	0	0	DATA BUS → ICW1
1	X	X	1	0	0	DATA BUS → OCW1, ICW2, ICW3, ICW4 (Note 2)
						DISABLE FUNCTION
X	X	X	1	1	0	DATA BUS — 3-STATE (NO OPERATION)
X	X	X	X	X	1	DATA BUS — 3-STATE (NO OPERATION)

Notes: 1. Selection of IRR, ISR or Interrupting Level is based on the content of OCW3 written before the READ operation.

2. On-chip sequencer logic queues these commands into proper sequence.

## 8259A Basic Operation



### INITIALIZATION COMMAND WORDS 1 AND 2 (ICW1, ICW2)

**A<sub>5</sub>-A<sub>15</sub>:** Page starting address of service routines. In an MCS 80/85 system, the 8 request levels will generate CALLS to 8 locations equally spaced in memory. These can be programmed to be spaced at intervals of 4 or 8 memory locations, thus the 8 routines will occupy a page of 32 or 64 bytes, respectively.

The address format is 2 bytes long (A<sub>0</sub>-A<sub>15</sub>). When the routine interval is 4, A<sub>0</sub>-A<sub>4</sub> are automatically inserted by the 8259A, while A<sub>5</sub>-A<sub>15</sub> are programmed externally. When the routine interval is 8, A<sub>0</sub>-A<sub>5</sub> are automatically inserted by the 8259A, while A<sub>6</sub>-A<sub>15</sub> are programmed externally.

The 8-byte interval will maintain compatibility with current software, while the 4-byte interval is best for a compact jump table.

In an MCS-86 system T7-T3 are inserted in the five most significant bits of the vectoring byte and the 8259A sets the three least significant bits according to the interrupt level. A<sub>10</sub>-A<sub>5</sub> are ignored and ADI (Address Interval) has no effect.

**LTIM:** If LTIM = 1, then the 8259A will operate in the level interrupt mode. Edge detect logic on the interrupt inputs will be disabled.

**ADI:** CALL address interval. ADI = 1 then interval = 4; ADI = 0 then interval = 8.

**SNGL:** Single. Means that this is the only 8259A in the system. If SNGL = 1 no ICW3 will be issued.

**IC4:** If this bit is set — ICW4 has to be read. If ICW4 is not needed, set IC4 = 0.

### INITIALIZATION COMMAND WORD 3 (ICW3)

This word is read only when there is more than one 8259A in the system and cascading is used, in which case SNGL = 0. It will load the 8-bit slave register. The functions of this register are:

- In the master mode (either when  $\overline{SP} = 1$ , or in buffered mode when  $M/S = 1$  in ICW4) a "1" is set for each slave in the system. The master then will release byte 1 of the call sequence (for MCS-80/85 system) and will enable the corresponding slave to release bytes 2 and 3 (for 8086/8088 only byte 2) through the cascade lines.
- In the slave mode (either when  $\overline{SP} = 0$ , or if BUF = 1 and  $M/S = 0$  in ICW4) bits 2-0 identify the slave. The slave compares its cascade input with these bits and if they are equal, bytes 2 and 3 of the call sequence (or just byte 2 for 8086/8088) are released by it on the Data Bus.

### INITIALIZATION COMMAND WORD 4 (ICW4)

**SFNM:** If SFNM = 1 the special fully nested mode is programmed.

**BUF:** If BUF = 1 the buffered mode is programmed. In buffered mode  $\overline{SP}/\overline{EN}$  becomes an enable output and the master/slave determination is by M/S.

**M/S:** If buffered mode is selected: M/S = 1 means the 8259A is programmed to be a master, M/S = 0 means the 8259A is programmed to be a slave. If BUF = 0, M/S has no function.

**AEOI:** If AEOI = 1 the automatic end of interrupt mode is programmed.

**$\mu$ PM:** Microprocessor mode:  $\mu$ PM = 0 sets the 8259A for MCS-80/85 system operation,  $\mu$ PM = 1 sets the 8259A for MCS-86 system operation.

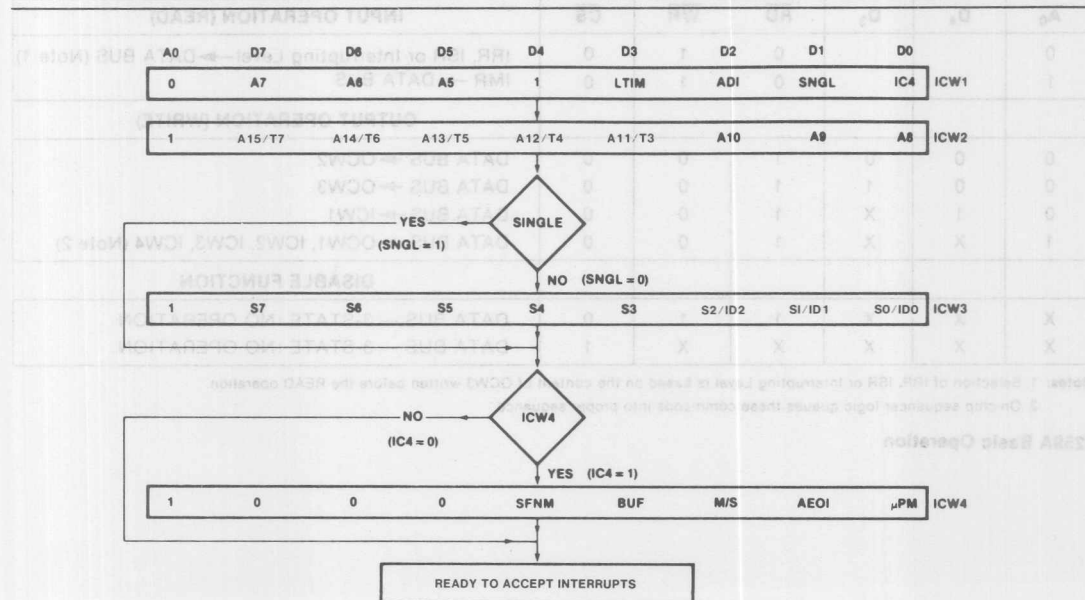
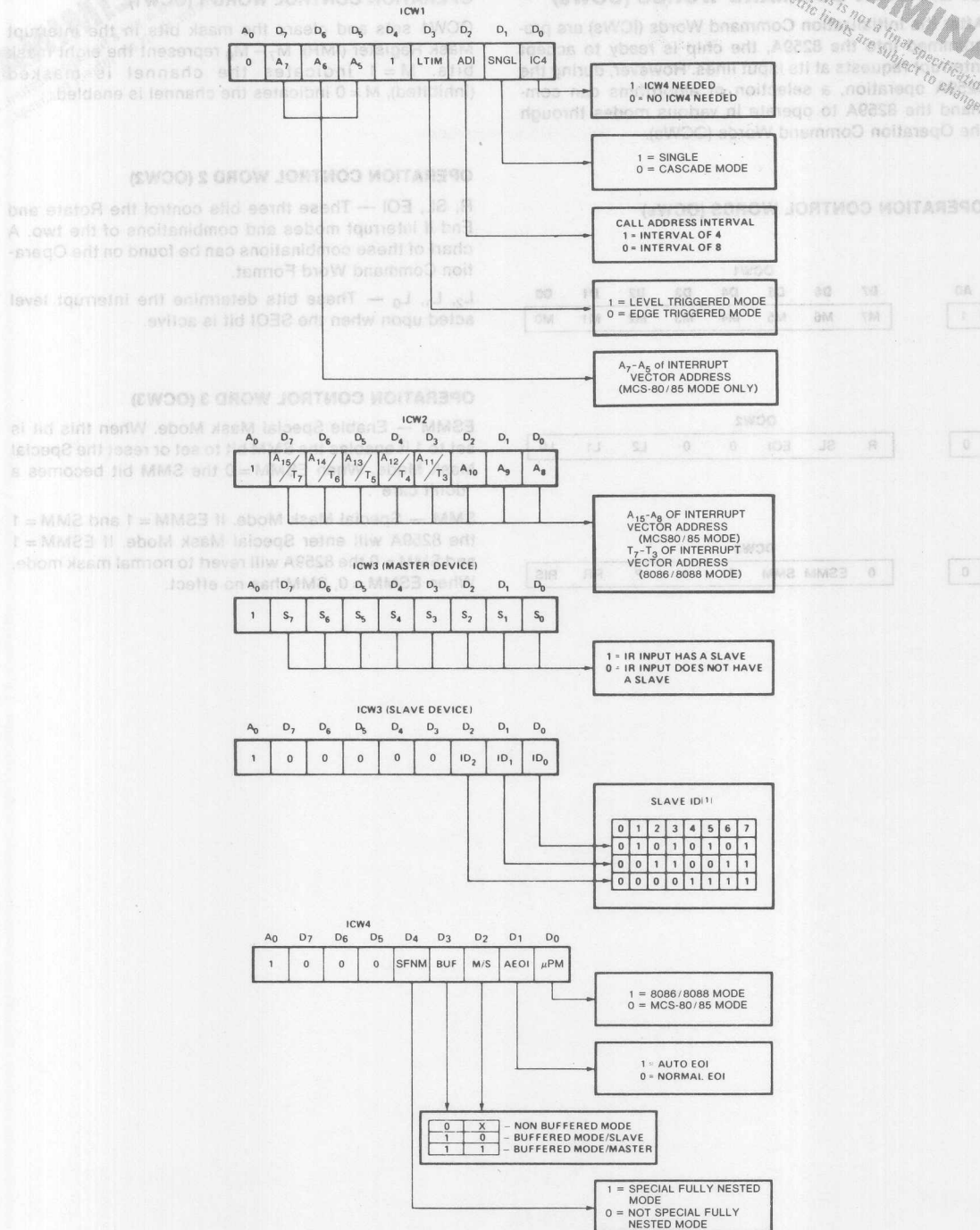


Figure 1. Initialization Sequence



**PRELIMINARY**  
 Notice: This is not a final specification. Some  
 parametric limits are subject to change.



NOTE 1: SLAVE ID IS EQUAL TO THE CORRESPONDING MASTER IR INPUT.

#### Initialization Command Word Format

**OPERATION COMMAND WORDS (OCWs)**

After the Initialization Command Words (ICWs) are programmed into the 8259A, the chip is ready to accept interrupt requests at its input lines. However, during the 8259A operation, a selection of algorithms can command the 8259A to operate in various modes through the Operation Command Words (OCWs).

**OPERATION CONTROL WORDS (OCWs)**

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

**OCW1**

A0	R	SL	EOI	0	0	L2	L1	L0
0								

**OCW2**

A0	0	ESMM	SMM	0	1	P	RR	RIS
0								

**OCW3****OPERATION CONTROL WORD 1 (OCW1)**

OCW1 sets and clears the mask bits in the interrupt Mask Register (IMR). M<sub>7</sub>–M<sub>0</sub> represent the eight mask bits. M = 1 indicates the channel is masked (inhibited), M = 0 indicates the channel is enabled.

**OPERATION CONTROL WORD 2 (OCW2)**

R, SL, EOI — These three bits control the Rotate and End if Interrupt modes and combinations of the two. A chart of these combinations can be found on the Operation Command Word Format.

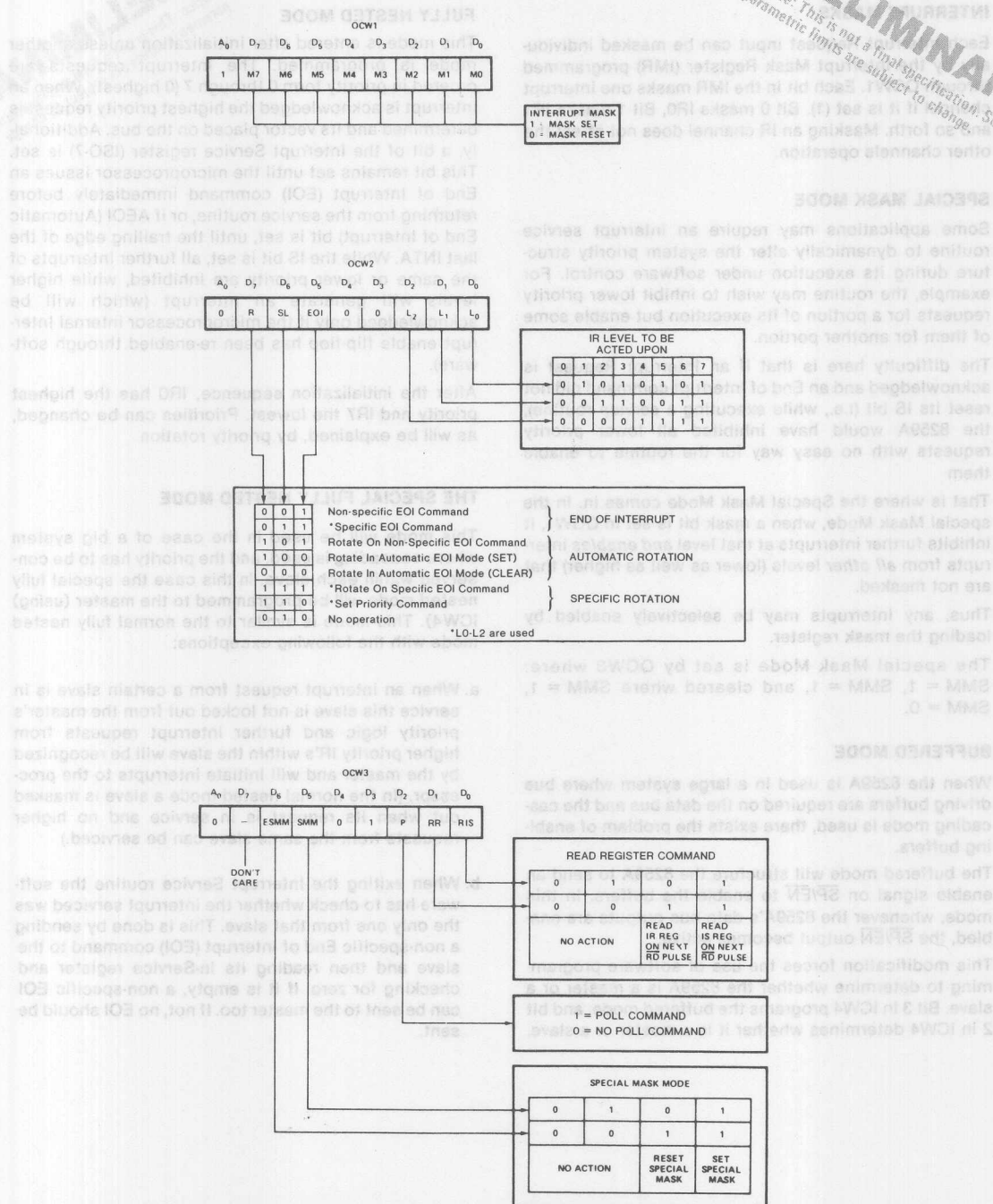
L<sub>2</sub>, L<sub>1</sub>, L<sub>0</sub> — These bits determine the interrupt level acted upon when the SEOI bit is active.

**OPERATION CONTROL WORD 3 (OCW3)**

ESMM — Enable Special Mask Mode. When this bit is set to 1 it enables the SMM bit to set or reset the Special Mask Mode. When ESMM = 0 the SMM bit becomes a "don't care".

SMM — Special Mask Mode. If ESMM = 1 and SMM = 1 the 8259A will enter Special Mask Mode. If ESMM = 1 and SMM = 0 the 8259A will revert to normal mask mode. When ESMM = 0, SMM has no effect.

**PRELIMINARY**  
 Notice: This is not a final specification. Some parametric limits are subject to change.



Operation Command Word Format

## INTERRUPT MASKS

Each Interrupt Request input can be masked individually by the Interrupt Mask Register (IMR) programmed through OCW1. Each bit in the IMR masks one interrupt channel if it is set (1). Bit 0 masks IR0, Bit 1 masks IR1 and so forth. Masking an IR channel does not affect the other channels operation.

## SPECIAL MASK MODE

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion.

The difficulty here is that if an Interrupt Request is acknowledged and an End of Interrupt command did not reset its IS bit (i.e., while executing a service routine), the 8259A would have inhibited all lower priority requests with no easy way for the routine to enable them

That is where the Special Mask Mode comes in. In the special Mask Mode, when a mask bit is set in OCW1, it inhibits further interrupts at that level *and enables* interrupts from *all other* levels (lower as well as higher) that are not masked.

Thus, any interrupts may be selectively enabled by loading the mask register.

The special Mask Mode is set by OCW3 where: SMM = 1, SMM = 1, and cleared where SMM = 1, SMM = 0.

## BUFFERED MODE

When the 8259A is used in a large system where bus driving buffers are required on the data bus and the cascading mode is used, there exists the problem of enabling buffers.

The buffered mode will structure the 8259A to send an enable signal on SP/EN to enable the buffers. In this mode, whenever the 8259A's data bus outputs are enabled, the SP/EN output becomes active.

This modification forces the use of software programming to determine whether the 8259A is a master or a slave. Bit 3 in ICW4 programs the buffered mode, and bit 2 in ICW4 determines whether it is a master or a slave.

## FULLY NESTED MODE

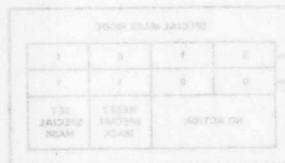
This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered in priority form 0 through 7 (0 highest). When an interrupt is acknowledged the highest priority request is determined and its vector placed on the bus. Additionally, a bit of the Interrupt Service register (ISO-7) is set. This bit remains set until the microprocessor issues an End of Interrupt (EOI) command immediately before returning from the service routine, or if AEOL (Automatic End of Interrupt) bit is set, until the trailing edge of the last INTA. While the IS bit is set, all further interrupts of the same or lower priority are inhibited, while higher levels will generate an interrupt (which will be acknowledged only if the microprocessor internal Interrupt enable flip-flop has been re-enabled through software).

After the initialization sequence, IR0 has the highest priority and IR7 the lowest. Priorities can be changed, as will be explained, by priority rotation.

## THE SPECIAL FULLY NESTED MODE

This mode will be used in the case of a big system where cascading is used, and the priority has to be conserved within each slave. In this case the special fully nested mode will be programmed to the master (using ICW4). This mode is similar to the normal fully nested mode with the following exceptions:

- When an interrupt request from a certain slave is in service this slave is not locked out from the master's priority logic and further interrupt requests from higher priority IR's within the slave will be recognized by the master and will initiate interrupts to the processor. (In the normal nested mode a slave is masked out when its request is in service and no higher requests from the same slave can be serviced.)
- When exiting the Interrupt Service routine the software has to check whether the interrupt serviced was the only one from that slave. This is done by sending a non-specific End of Interrupt (EOI) command to the slave and then reading its In-Service register and checking for zero. If it is empty, a non-specific EOI can be sent to the master too. If not, no EOI should be sent.



## POLL

In this mode the microprocessor internal Interrupt Enable flip-flop is reset, disabling its interrupt input. Service to devices is achieved by programmer initiative using a Poll command.

The Poll command is issued by setting  $P = "1"$  in OCW3. The 8259A treats the next  $\overline{RD}$  pulse to the 8259A (i.e.,  $\overline{RD} = 0$ ,  $\overline{CS} = 0$ ) as an interrupt acknowledge, sets the appropriate IS bit if there is a request, and reads the priority level. Interrupt is frozen from  $\overline{WR}$  to  $\overline{RD}$ .

The word enabled onto the data bus during  $\overline{RD}$  is:

D7	D6	D5	D4	D3	D2	D1	D0
I	—	—	—	—	W2	W1	W0

W0-W2: Binary code of the highest priority level requesting service.

I: Equal to a "1" if there is an interrupt.

This mode is useful if there is a routine command common to several levels so that the INTA sequence is not needed (saves ROM space). Another application is to use the poll command to expand the number of priority levels to more than 64.

## END OF INTERRUPT (EOI)

The In Service (IS) bit can be reset either automatically following the trailing edge of the last in sequence INTA pulse (when AEOL bit in ICW1 is set) or by a command word that must be issued to the 8259A before returning from a service routine (EOI command). An EOI command must be issued twice, once for the master and once for the corresponding slave if slaves are in use.

There are two forms of EOI command: Specific and Non-Specific. When the 8259A is operated in modes which preserve the fully nested structure, it can determine which IS bit to reset on EOI. When a Non-Specific EOI command is issued the 8259A will automatically reset the highest IS bit of those that are set, since in the nested mode the highest IS level was necessarily the last level acknowledged and serviced.

However, when a mode is used which may disturb the fully nested structure, the 8259A may no longer be able to determine the last level acknowledged. In this case a Specific End of Interrupt (SEOI) must be issued which includes as part of the command the IS level to be reset. EOI is issued whenever  $EOI = 1$ , in OCW2, where L0-L2 is the binary level of the IS bit to be reset. Note that although the Rotate command can be issued together with an EOI where  $EOI = 1$ , it is not necessarily tied to it.

It should be noted that an IS bit that is masked by an IMR bit will not be cleared by a non-specific EOI if the 8259A is in the Special Mask Mode.

## AUTOMATIC END OF INTERRUPT (AEOL) MODE

If  $AEOL = 1$  in ICW4, then the 8259A will operate in AEOL mode continuously until reprogrammed by ICW4. In this mode the 8259A will automatically perform a non-specific EOI operation at the trailing edge of the last interrupt acknowledge pulse (third pulse in MCS-80/85,

second in MCS-86). Note that from a system standpoint, this mode should be used only when a nested multilevel interrupt structure is not required within a single 8259A.

To achieve automatic rotation within AEOL, there is a special rotate flip-flop. It is set by OCW2 with  $R = 1$ ,  $SL = 0$ .  $EOI = 0$ , and cleared with  $R = 0$ ,  $SEOI = 0$ ,  $EOI = 0$ .

## AUTOMATIC ROTATION (Equal Priority Devices)

In some applications there are a number of interrupting devices of equal priority. In this mode a device, after being serviced, receives the lowest priority, so a device requesting an interrupt will have to wait, in the worst case until each of 7 other devices are serviced at most once. For example, if the priority and "in service" status is:

**Before Rotate** (IR4 the highest priority requiring service)

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	1	0	1	0	0	0	0
Lowest Priority				Highest Priority			
7	6	5	4	3	2	1	0

**After Rotate** (IR4 was serviced, all other priorities rotated correspondingly)

IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
0	1	0	0	0	0	0	0
Highest Priority				Lowest Priority			
2	1	0	7	6	5	4	3

There are two ways to accomplish Automatic Rotation using OCW2, the Rotate on Non-Specific EOI Command ( $R = 1$ ,  $SL = 0$ ,  $EOI = 1$ ) and the Rotate in Automatic EOI Mode which is set by ( $R = 1$ ,  $SL = 0$ ,  $EOI = 0$ ) and cleared by ( $R = 0$ ,  $SL = 0$ ,  $EOI = 0$ ).

## SPECIFIC ROTATION (Specific Priority)

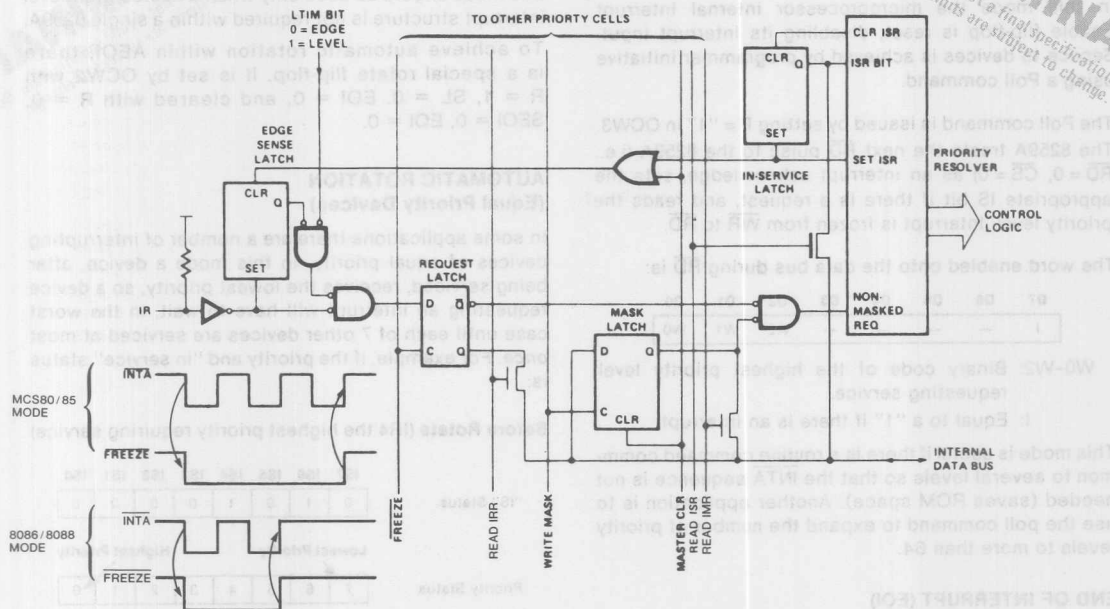
The programmer can change priorities by programming the bottom priority and thus fixing all other priorities; i.e., if IR5 is programmed as the bottom priority device, then IR6 will have the highest one.

The Set Priority command is issued in OCW2 where:  $R = 1$ ,  $SEOI = 1$ ; L0-L2 is the binary priority level code of the bottom priority device.

Observe that in this mode internal status is updated by software control during OCW2. However, it is independent of the End of Interrupt (EOI) command (also executed by OCW2). Priority changes can be executed during an EOI command by using the Rotate on Specific EOI Command in OCW2 ( $R = 1$ ,  $SL = 1$ ,  $EOI = 1$  and L0-L2 = IR level to receive bottom priority).



**PRELIMINARY**  
 Notice: This is not a final specification. Some parametric limits are subject to change.



## NOTES

1. MASTER CLEAR ACTIVE ONLY DURING ICW1
2. FREEZE/IS ACTIVE DURING INTA/ AND POLL SEQUENCES ONLY
3. TRUTH TABLE FOR D-LATCH

C	D	Q	OPERATION
1	Di	Di	FOLLOW
0	X	Qn-1	HOLD

## Priority Cell — Simplified Logic Diagram

## LEVEL TRIGGERED MODE

This mode is programmed using bit 3 in ICW1.

If LTIM = '1,' an interrupt request will be recognized by a 'high' level on IR Input, and there is no need for an edge detection. The interrupt request must be removed before the EOI command is issued or the CPU interrupt is enabled to prevent a second interrupt from occurring.

The above figure shows a conceptual circuit to give the reader an understanding of the level sensitive and edge sensitive input circuitry of the 8259A. Be sure to note that the request latch is a transparent D type latch.

## READING THE 8259A STATUS

The input status of several internal registers can be read to update the user information on the system. The following registers can be read via OCW3 (IRR and ISR or OCW1 (IMR).

**Interrupt Request Register (IRR):** 8-bit register which contains the levels requesting an interrupt to be acknowledged. The highest request level is reset from the IRR when an interrupt is acknowledged. (Not affected by IMR).

**In-Service Register (ISR):** 8-bit register which contains the priority levels that are being serviced. The ISR is updated when an End of Interrupt command is issued.

**Interrupt Mask Register:** 8-bit register which contains the interrupt request lines which are masked.

The IRR can be read when, prior to the  $\overline{RD}$  pulse, a Read Register Command is issued with OCW3 (RR = 1, RIS = 0).

The ISR can be read when, prior to the  $\overline{RD}$  pulse, a Read Register Command is issued with OCW3 (RR = 1, RIS = 1).

There is no need to write an OCW3 before every status read operation, as long as the status read corresponds with the previous one; i.e., the 8259A "remembers" whether the IRR or ISR has been previously selected by the OCW3. This is not true when poll is used.

After initialization the 8259A is set to IRR.

For reading the IMR, no OCW3 is needed. The output data bus will contain the IMR whenever  $\overline{RD}$  is active and A0 = 1 (OCW1).

Polling overrides status read when P = 1, RR = 1 in OCW3.

## SUMMARY OF 8259A INSTRUCTION SET

Inst. #	Mnemonic	A0	D7	D6	D5	D4	D3	D2	D1	D0	Operation Description
1	ICW1 A	0	A7	A6	A5	1	0	1	1	0	Format = 4, single, edge triggered
2	ICW1 B	0	A7	A6	A5	1	1	1	1	0	Format = 4, single, level triggered
3	ICW1 C	0	A7	A6	A5	1	0	1	0	0	Format = 4, not single, edge triggered
4	ICW1 D	0	A7	A6	A5	1	1	1	0	0	Format = 4, not single, level triggered
5	ICW1 E	0	A7	A6	0	1	0	0	1	0	Format = 8, single, edge triggered
6	ICW1 F	0	A7	A6	0	1	1	0	1	0	Format = 8, single, level triggered
7	ICW1 G	0	A7	A6	0	1	0	0	0	0	Format = 8, not single, edge triggered
8	ICW1 H	0	A7	A6	0	1	1	0	0	0	Format = 8, not single, level triggered
9	ICW1 I	0	A7	A6	A5	1	0	1	1	1	Format = 4, single, edge triggered
10	ICW1 J	0	A7	A6	A5	1	1	1	1	1	Format = 4, single, level triggered
11	ICW1 K	0	A7	A6	A5	1	0	1	0	1	Format = 4, not single, edge triggered
12	ICW1 L	0	A7	A6	A5	1	1	1	0	1	Format = 4, not single, level triggered
13	ICW1 M	0	A7	A6	0	1	0	0	1	1	Format = 8, single, edge triggered
14	ICW1 N	0	A7	A6	0	1	1	0	1	1	Format = 8, single, level triggered
15	ICW1 O	0	A7	A6	0	1	0	0	0	1	Format = 8, not single, edge triggered
16	ICW1 P	0	A7	A6	0	1	1	0	0	1	Format = 8, not single, level triggered
17	ICW2	1	A15	A14	A13	A12	A11	A10	A9	A8	Byte 2 initialization
18	ICW3 M	1	S7	S6	S5	S4	S3	S2	S1	S0	Byte 3 initialization — master
19	ICW3 S	1	0	0	0	0	0	S2	S1	S0	Byte 3 initialization — slave
20	ICW4 A	1	0	0	0	0	0	0	0	0	No action, redundant
21	ICW4 B	1	0	0	0	0	0	0	0	1	Non-buffered mode, no AEOI, 8086/8088
22	ICW4 C	1	0	0	0	0	0	0	1	0	Non-buffered mode, AEOI, MCS-80/85
23	ICW4 D	1	0	0	0	0	0	0	1	1	Non-buffered mode, AEOI, 8086/8088
24	ICW4 E	1	0	0	0	0	0	0	1	0	No action, redundant
25	ICW4 F	1	0	0	0	0	0	0	1	0	Non-buffered mode, no AEOI, 8086/8088
26	ICW4 G	1	0	0	0	0	0	0	1	1	Non-buffered mode, AEOI, MCS-80/85
27	ICW4 H	1	0	0	0	0	0	0	1	1	Non-buffered mode, AEOI, 8086/8088
28	ICW4 I	1	0	0	0	0	0	1	0	0	Buffered mode, slave, no AEOI, MCS-80/85
29	ICW4 J	1	0	0	0	0	0	1	0	0	Buffered mode, slave, no AEOI, 8086/8088
30	ICW4 K	1	0	0	0	0	0	1	0	1	Buffered mode, slave, AEOI, MCS-80/85
31	ICW4 L	1	0	0	0	0	0	1	0	1	Buffered mode, slave, AEOI, 8086/8088
32	ICW4 M	1	0	0	0	0	0	1	1	0	Buffered mode, master, no AEOI, MCS-80/85
33	ICW4 N	1	0	0	0	0	0	1	1	0	Buffered mode, master, no AEOI, 8086/8088
34	ICW4 O	1	0	0	0	0	0	1	1	1	Buffered mode, master, AEOI, MCS-80/85
35	ICW4 P	1	0	0	0	0	0	1	1	1	Buffered mode, master AEOI, 8086, 8088
36	ICW4 NA	1	0	0	0	1	0	0	0	0	Fully nested mode, MCS-80, non buffered, no AEOI
37	ICW4 NB	1	0	0	0	1	0	0	0	1	ICW4 NB through ICW4 ND are identical to ICW4 B through ICW4 D with the addition of Fully Nested Mode
38	ICW4 NC	1	0	0	0	1	0	0	1	0	
39	ICW4 ND	1	0	0	0	1	0	0	1	1	
40	ICW4 NE	1	0	0	0	1	0	1	0	0	
41	ICW4 NF	1	0	0	0	1	0	1	0	1	Fully Nested Mode, MCS-80/85, non-buffered, no AEOI
42	ICW4 NG	1	0	0	0	1	0	1	1	0	
43	ICW4 NH	1	0	0	0	1	0	1	1	1	
44	ICW4 NI	1	0	0	0	1	1	0	0	0	
45	ICW4 NJ	1	0	0	0	1	1	0	0	1	ICW4 NF through ICW4 NP are identical to ICW4 F through ICW4 P with the addition of Fully Nested Mode
46	ICW4 NK	1	0	0	0	1	1	0	1	0	
47	ICW4 NL	1	0	0	0	1	1	0	1	1	
48	ICW4 NM	1	0	0	0	1	1	1	0	0	
49	ICW4 NN	1	0	0	0	1	1	1	0	1	
50	ICW4 NO	1	0	0	0	1	1	1	1	0	
51	ICW4 NP	1	0	0	0	1	1	1	1	1	
52	OCW1	1	M7	M6	M5	M4	M3	M2	M1	M0	Load mask register, read mask register
53	OCW2 E	0	0	0	1	0	0	0	0	0	Non-specific EOI
54	OCW2 SE	0	0	1	1	0	0	L2	L1	L0	Specific EOI, L0-L2 code of IS FF to be reset
55	OCW2 RE	0	1	0	1	0	0	0	0	0	Rotate on Non-Specific EOI
56	OCW2 RSE	0	1	1	1	0	0	L2	L1	L0	Rotate on Specific EOI L0-L2 code of line
57	OCW2 R	0	1	0	0	0	0	0	0	0	Rotate in Auto EOI (set)
58	OCW2 CR	0	0	0	0	0	0	0	0	0	Rotate in Auto EOI (clear)
59	OCW2 RS	0	1	1	0	0	0	L2	L1	L0	Set Priority Command
60	OCW3 P	0	0	0	0	0	1	1	0	0	Poll mode
61	OCW3 RIS	0	0	0	0	0	1	0	1	1	Read IS register

Notice: This is not a final specification. Some parametric limits are subject to change.

Notice: This is not a final specification. Some parametric limits are subject to change.

Notice: This is not a final specification. Some parametric limits are subject to change.

Notice: This is not a final specification. Some parametric limits are subject to change.

Notice: This is not a final specification. Some parametric limits are subject to change.

Notice: This is not a final specification. Some parametric limits are subject to change.

Notice: This is not a final specification. Some parametric limits are subject to change.

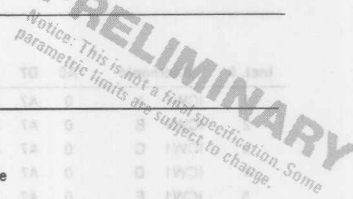
Notice: This is not a final specification. Some parametric limits are subject to change.

Notice: This is not a final specification. Some parametric limits are subject to change.

**RELIMINARY**

Notice: This is not a final specification. Some parametric limits are subject to change.

TO	FROM	LAND
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0



**RELIMINARY**

Notice: This is not a final specification. Some parametric limits are subject to change.

TO	FROM	LAND
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0

## PIN FUNCTIONS

NAME	I/O	PIN#	FUNCTION	NAME	I/O	PIN#	FUNCTION
V <sub>CC</sub>	I	28	+5v supply	INT	O	17	Interrupt: This pin goes high whenever a valid interrupt request is asserted. It is used to interrupt the CPU, thus it is connected to the CPU's interrupt pin.
GND	I	14	Ground	IR <sub>0</sub> -IR <sub>7</sub>	I	18-25	Interrupt Requests: Asynchronous inputs. An interrupt request can be generated by raising an IR input (low to high) and holding it high until it is acknowledged (Edge Triggered Mode), or just by a high level on an IR input (Level Triggered Mode).
CS	I	1	Chip Select A low on this pin enables RD and WR communication between the CPU and the 8259A. INTA functions are independent of CS.	INTA	I	26	Interrupt Acknowledge: This pin is used to enable 8259A interrupt-vector data onto the data bus. This is done by a sequence of interrupt acknowledge pulses issued by the CPU.
WR	I	2	Write: A low on this pin when CS is low, enables the 8259A to accept command words from the CPU.	A <sub>0</sub>	I	27	AO Address Line: This pin acts in conjunction with the CS, WR, and RD pins. It is used by the 8259A to decipher between various Command Words the CPU writes and status the CPU wishes to read. It is typically connected to the CPU A0 address line (A1 for 8086/8088).
RD	I	3	Read: A low on this pin when CS is low enables the 8259A to release status onto the data bus for the CPU.				
D <sub>7</sub> -D <sub>0</sub>	I/O	4-11	Bidirectional Data Bus: Control, status and interrupt-vector information is transferred via this bus.				
CAS <sub>0</sub> -CAS <sub>2</sub>	I/O	12,13,15	Cascade Lines: The CAS lines form a private 8259A bus to control a multiple 8259A structure. These pins are outputs for a master 8259A and inputs for a slave 8259A.				
SP/EN	I/O	16	Slave Program/Enable Buffer: This is a dual function pin. When in the Buffered Mode it can be used as an output to control buffer transceivers (EN). When not in the buffered mode it is used as an input to designate a master (SP = 1) or slave (SP = 0).				

## ABSOLUTE MAXIMUM RATINGS\*

Ambient Temperature Under Bias . . . . . -40°C to 85°C  
Storage Temperature . . . . . -65°C to +150°C  
Voltage On Any Pin

With Respect to Ground . . . . . -0.5V to +7V  
Power Dissipation . . . . . 1 Watt

## \*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

## D.C. CHARACTERISTICS

T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = 5V ± 10% (8259-A), V<sub>CC</sub> = 5V ± 10% (8259A)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V <sub>IL</sub>	Input Low Voltage	-.5	V		
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub> + .5V	V	
V <sub>OL</sub>	Output Low Voltage		.45	V	I <sub>OL</sub> = 2.2 mA
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -400 μA
V <sub>OH(INT)</sub>	Interrupt Output High Voltage	3.5 2.4		C V	I <sub>OH</sub> = -100 μA I <sub>OH</sub> = -400 μA
I <sub>LI</sub>	Input Load Current		10	μA	V <sub>IN</sub> = V <sub>CC</sub> to 0V
I <sub>LOL</sub>	Output Leakage Current		-10	μA	V <sub>OUT</sub> = 0.45V
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		85	mA	
I <sub>LIR</sub>	IR Input Load Current		-300 10	μA μA	V <sub>IN</sub> = 0 V <sub>IN</sub> = V <sub>CC</sub>

## 8259A A.C. CHARACTERISTICS

 $T_A = 0^\circ\text{C to } 70^\circ\text{C}$   $V_{CC} = 5V \pm 5\%$  (8259A-8)  $V_{CC} = 5V \pm 10\%$  (8259A)

## TIMING REQUIREMENTS

Symbol	Parameter	8259A-8		8259A		8259A-2		Units	Test Conditions
		Min.	Max.	Min.	Max.	Min.	Max.		
TAHRL	AO/ $\overline{\text{CS}}$ Setup to $\overline{\text{RD}}/\overline{\text{INTA}}$	50		0		0		ns	
TRHAX	AO/ $\overline{\text{CS}}$ Hold after $\overline{\text{RD}}/\overline{\text{INTA}}$	5		0		0		ns	
TRLRH	$\overline{\text{RD}}$ Pulse Width	420		235		160		ns	
TAHWL	AO/ $\overline{\text{CS}}$ Setup to $\overline{\text{WR}}$	50		0		0		ns	
TWHAX	AO/ $\overline{\text{CS}}$ Hold after $\overline{\text{WR}}$	20		0		0		ns	
TWLWH	$\overline{\text{WR}}$ Pulse Width	400		290		190		ns	
TDVWH	Data Setup to $\overline{\text{WR}}$	300		240		160		ns	
TWHDX	Data Hold after $\overline{\text{WR}}$	40		0		0		ns	
TJLJH	Interrupt Request Width (Low)	100		100		100		ns	See Note 1
TCVIAL	Cascade Setup to Second or Third $\overline{\text{INTA}}$ (Slave Only)	55		55		40		ns	
TRHRL	End of $\overline{\text{RD}}$ to Next Command	160		160		160		ns	
TWHRL	End of $\overline{\text{WR}}$ to Next Command	190		190		190		ns	

Note: This is the low time required to clear the input latch in the edge triggered mode.

## TIMING RESPONSES

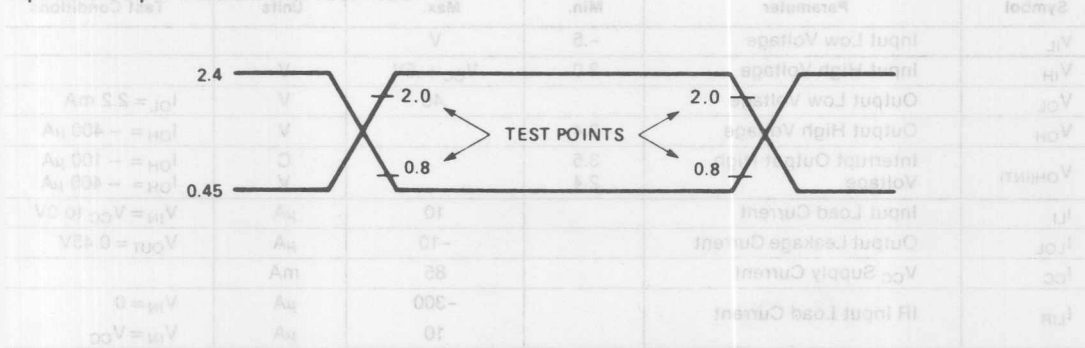
Symbol	Parameter	8259A-8		8259A		8259A-2		Units	Test Conditions
		Min.	Max.	Min.	Max.	Min.	Max.		
TRLDV	Data Valid from $\overline{\text{RD}}/\overline{\text{INTA}}$		300		200		120	ns	C of Data Bus = 100 pF
TRHDZ	Data Float after $\overline{\text{RD}}/\overline{\text{INTA}}$	10	200	100		85		ns	C of Data Bus Max test C = 100 pF Min. test C = 15 pF
TJHIH	Interrupt Output Delay		400		350		300	ns	C <sub>INT</sub> = 100 pF
TIAHCV	Cascade Valid from First $\overline{\text{INTA}}$ (Master Only)		565		565		360	ns	C <sub>CASCADE</sub> = 100 pF
TRLEL	Enable Active from $\overline{\text{RD}}$ or $\overline{\text{INTA}}$		160		125		100	ns	
TRHEH	Enable Inactive from $\overline{\text{RD}}$ or $\overline{\text{INTA}}$		325		150		150	ns	
TAHDV	Data Valid from Stable Address		350		200		200	ns	
TCVDV	Cascade Valid to Valid Data		300		300		200	ns	

## CAPACITANCE

 $T_A = 25^\circ\text{C}$ ;  $V_{CC} = \text{GND} = 0V$ 

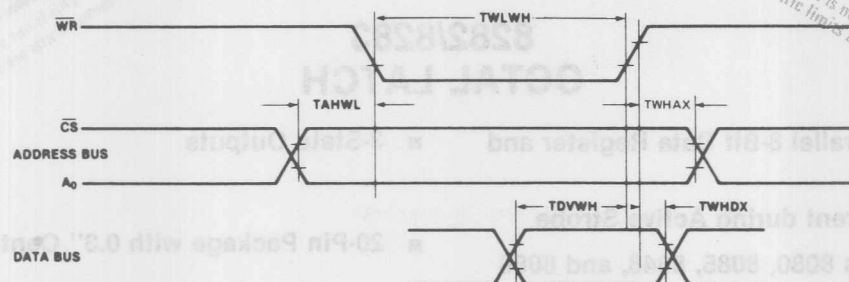
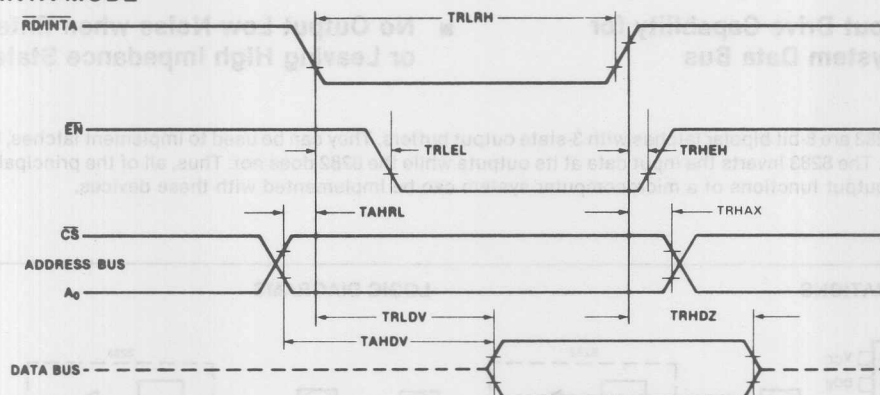
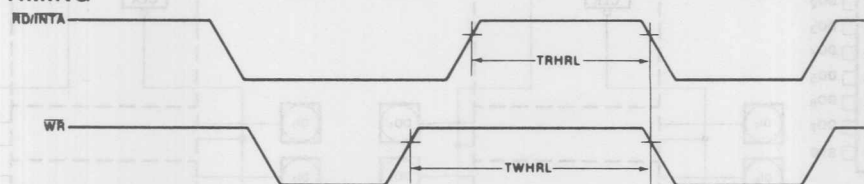
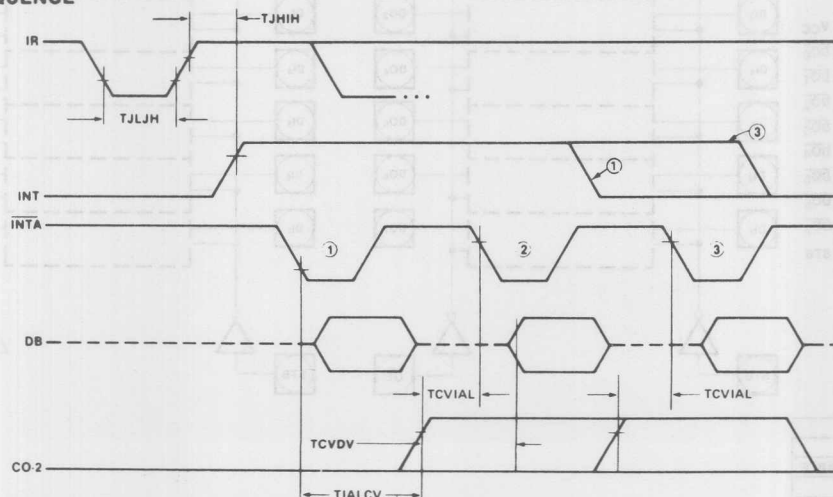
Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C <sub>IN</sub>	Input Capacitance			10	pF	f <sub>c</sub> = 1 MHz
C <sub>I/O</sub>	I/O Capacitance			20	pF	Unmeasured pins returned to V <sub>SS</sub>

## Input and Output Waveforms for A.C. Tests





**PRELIMINARY**  
 Notice: This is not a final specification. Some parametric limits are subject to change.

**WRITE MODE****READ/INTA MODE****OTHER TIMING****INTA SEQUENCE**

NOTE: Interrupt output must remain HIGH (at least) until leading edge of first INTA.  
 ① MCS 8088 Systems only  
 ② Cycle 1 in MCS 88 Systems, the Data Bus is not active

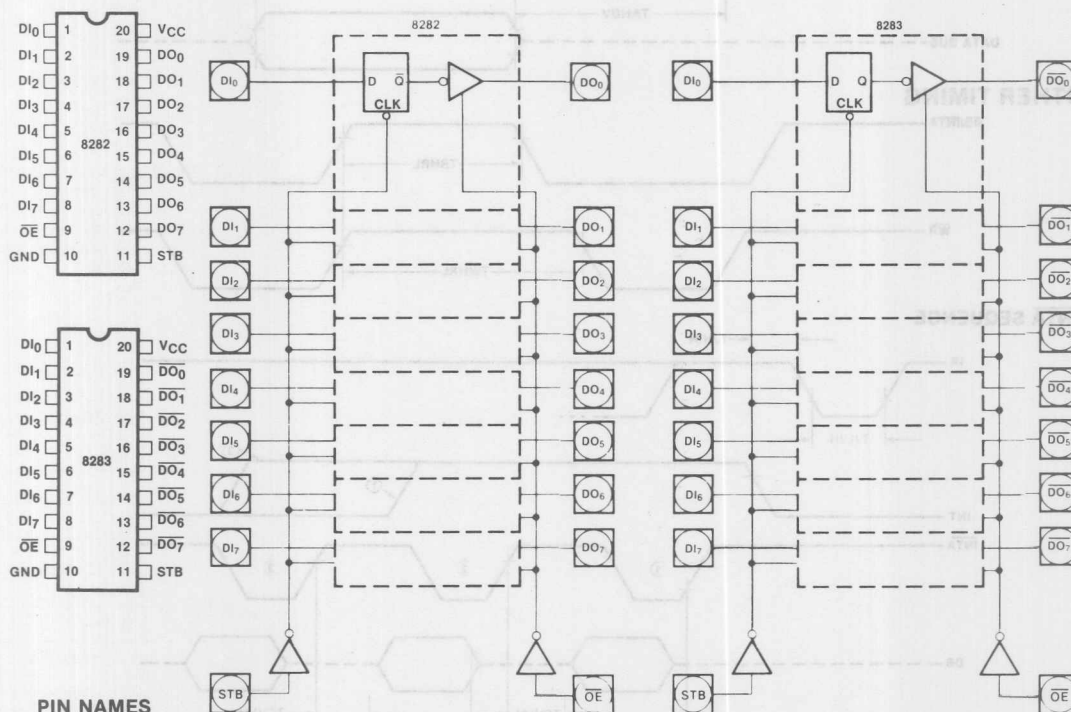
# 8282/8283 OCTAL LATCH

- Fully Parallel 8-Bit Data Register and Buffer
- 3-State Outputs
- Transparent during Active Strobe
- Supports 8080, 8085, 8048, and 8086 Systems
- 20-Pin Package with 0.3" Center
- High Output Drive Capability for Driving System Data Bus
- No Output Low Noise when Entering or Leaving High Impedance State

The 8282 and 8283 are 8-bit bipolar latches with 3-state output buffers. They can be used to implement latches, buffers, or multiplexers. The 8283 inverts the input data at its outputs while the 8282 does not. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with these devices.

## PIN CONFIGURATIONS

## LOGIC DIAGRAMS



## PIN NAMES

DI <sub>0</sub> -DI <sub>7</sub>	DATA IN
DO <sub>0</sub> -DO <sub>7</sub>	DATA OUT
OE	OUTPUT ENABLE
STB	STROBE

## PIN DEFINITIONS

Pin	Description
STB	STROBE (Input). STB is an input control pulse used to strobe data at the data input pins (A <sub>0</sub> -A <sub>7</sub> ) into the data latches. This signal is active HIGH to admit input data. The data is latched at the HIGH to LOW transition of STB.
$\overline{OE}$	OUTPUT ENABLE (Input). $\overline{OE}$ is an input control signal which when active LOW enables the contents of the data latches onto the data output pin (B <sub>0</sub> -B <sub>7</sub> ). $\overline{OE}$ being inactive HIGH forces the output buffers to their high impedance state.
DI <sub>0</sub> -DI <sub>7</sub>	DATA INPUT PINS (Input). Data presented at these pins satisfying setup time requirements when STB is strobed and latched into the data input latches.

DO<sub>0</sub>-DO<sub>7</sub> (8282) DATA OUTPUT PINS (Output). When  $\overline{OE}$  is true, the data in the data latches is presented as inverted (8283) or non-inverted (8282) data onto the data output pins.

## OPERATIONAL DESCRIPTION

The 8282 and 8283 octal latches are 8-bit latches with 3-state output buffers. Data having satisfied the setup time requirements is latched into the data latches by strobing the STB line HIGH to LOW. Holding the STB line in its active HIGH state makes the latches appear transparent. Data is presented to the data output pins by activating the  $\overline{OE}$  input line. When  $\overline{OE}$  is inactive HIGH the output buffers are in their high impedance state. Enabling or disabling the output buffers will not cause negative-going transients to appear on the data output bus.

## D.C. AND OPERATING CHARACTERISTICS

## ABSOLUTE MAXIMUM RATINGS\*

Temperature Under Bias . . . . . 0°C to 70°C  
 Storage Temperature . . . . . - 65°C to + 150°C  
 All Output and Supply Voltages . . . . . - 0.5V to + 7V  
 All Input Voltages . . . . . - 1.0V to + 5.5V  
 Power Dissipation . . . . . 1 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

## D.C. CHARACTERISTICS FOR 8282/8283

Conditions: V<sub>CC</sub> = 5V ± 5%, T<sub>A</sub> = 0°C to 70°C

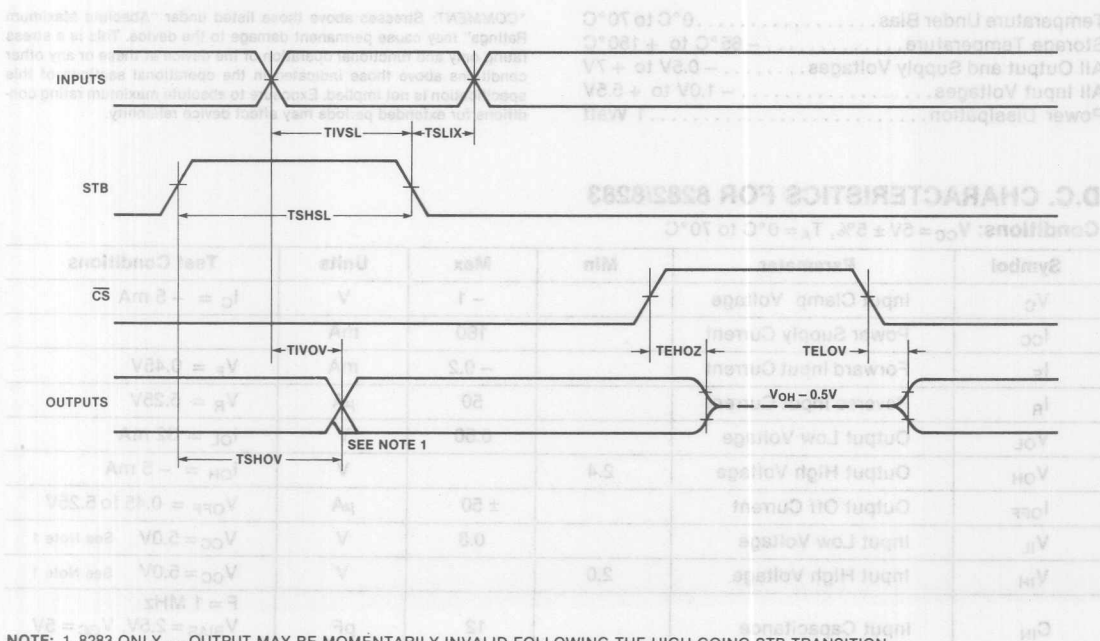
Symbol	Parameter	Min	Max	Units	Test Conditions
V <sub>C</sub>	Input Clamp Voltage		- 1	V	I <sub>C</sub> = - 5 mA
I <sub>CC</sub>	Power Supply Current		160	mA	
I <sub>F</sub>	Forward Input Current		- 0.2	mA	V <sub>F</sub> = 0.45V
I <sub>R</sub>	Reverse Input Current		50	μA	V <sub>R</sub> = 5.25V
V <sub>OL</sub>	Output Low Voltage		0.50	V	I <sub>OL</sub> = 32 mA
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = - 5 mA
I <sub>OFF</sub>	Output Off Current		± 50	μA	V <sub>OFF</sub> = 0.45 to 5.25V
V <sub>IL</sub>	Input Low Voltage		0.8	V	V <sub>CC</sub> = 5.0V See Note 1
V <sub>IH</sub>	Input High Voltage	2.0		V	V <sub>CC</sub> = 5.0V See Note 1
C <sub>IN</sub>	Input Capacitance		12	pF	F = 1 MHz V <sub>BIAS</sub> = 2.5V, V <sub>CC</sub> = 5V T <sub>A</sub> = 25°C

Notes: 1. Output Loading I<sub>OL</sub> = 32 mA, I<sub>OH</sub> = - 5 mA, C<sub>L</sub> = 300 pF

Symbol	Parameter	Min	Max	Units	Test Conditions
TIVOV	Input to Output Delay — Inverting — Non-Inverting		25	ns	(See Note 1)
			35	ns	
TSHOV	STB to Output Delay — Inverting — Non-Inverting		45	ns	
			55	ns	
TEHOZ	Output Disable Time		25	ns	
TELOV	Output Enable Time	10	50	ns	
TIVSL	Input to STB Setup Time	0		ns	
TSlix	Input to STB Hold Time	25		ns	
TSHSL	STB High Time	15		ns	

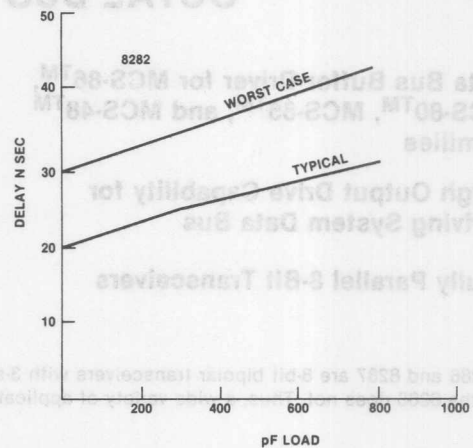
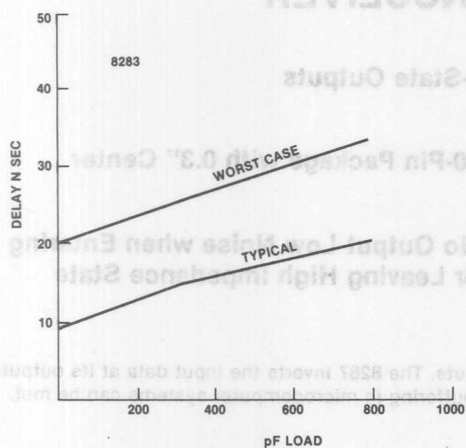
NOTE: 1. See waveforms and test load circuit on following page.

## 8282/8283 TIMING

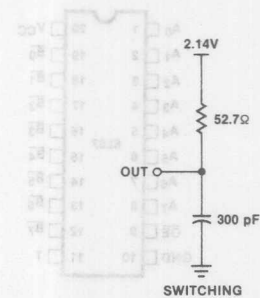
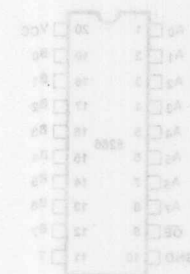
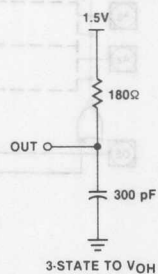
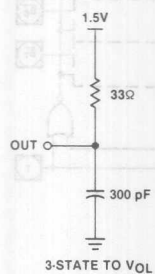


NOTE: 1. 8283 ONLY — OUTPUT MAY BE MOMENTARILY INVALID FOLLOWING THE HIGH GOING STB TRANSITION.

2. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS OTHERWISE NOTED



### OUTPUT TEST LOAD CIRCUITS



LOCAL BUS DATA	8282
SYSTEM BUS DATA	8283
OUTPUT ENABLE	8283
TRANSMIT	8283

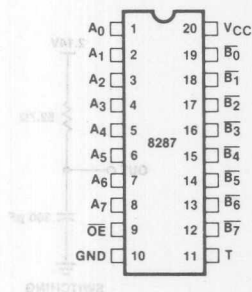
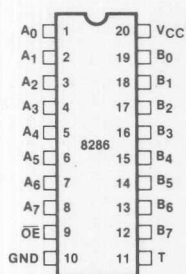


# 8286/8287 OCTAL BUS TRANSCEIVER

- Data Bus Buffer Driver for MCS-86™, MCS-80™, MCS-85™, and MCS-48™ Families
- High Output Drive Capability for Driving System Data Bus
- Fully Parallel 8-Bit Transceivers
- 3-State Outputs
- 20-Pin Package with 0.3" Center
- No Output Low Noise when Entering or Leaving High Impedance State

The 8286 and 8287 are 8-bit bipolar transceivers with 3-state outputs. The 8287 inverts the input data at its outputs while the 8286 does not. Thus, a wide variety of applications for buffering in microcomputer systems can be met.

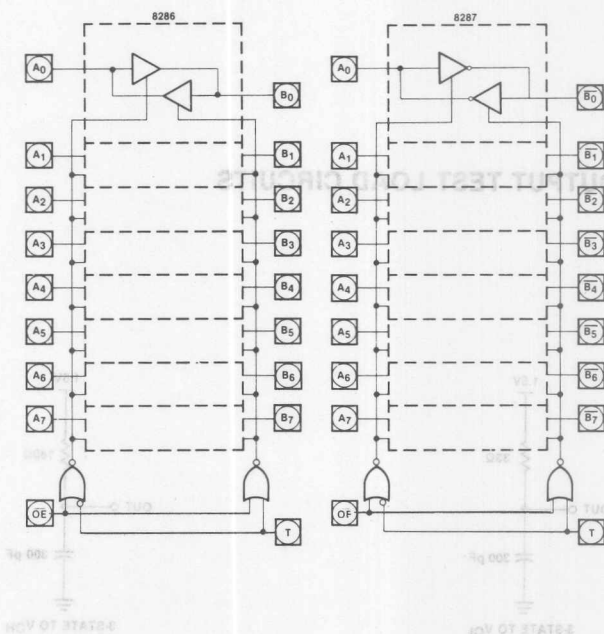
## PIN CONFIGURATIONS



## PIN NAMES

A <sub>0</sub> -A <sub>7</sub>	LOCAL BUS DATA
B <sub>0</sub> -B <sub>7</sub>	SYSTEM BUS DATA
OE	OUTPUT ENABLE
T	TRANSMIT

## LOGIC DIAGRAMS



## PIN DEFINITIONS

Pin	Description
T	TRANSMIT (Input). T is an input control signal used to control the direction of the transceivers. When HIGH, it configures the transceiver's B <sub>0</sub> -B <sub>7</sub> as outputs with A <sub>0</sub> -A <sub>7</sub> as inputs. T LOW configures A <sub>0</sub> -A <sub>7</sub> as the outputs with B <sub>0</sub> -B <sub>7</sub> serving as the inputs.
$\overline{OE}$	OUTPUT ENABLE (Input). $\overline{OE}$ is an input control signal used to enable the appropriate output driver (as selected by T) onto its respective bus. This signal is active LOW.
A <sub>0</sub> -A <sub>7</sub>	LOCAL BUS DATA PINS (Input/Output). These pins serve to either present data to or accept data from the processor's local bus depending upon the state of the T pin.

B<sub>0</sub>-B<sub>7</sub>  
(8286)  
B<sub>0</sub>-B<sub>7</sub>  
(8287)

SYSTEM BUS DATA PINS (Input/Output). These pins serve to either present data to or accept data from the system bus depending upon the state of the T pin.

## OPERATIONAL DESCRIPTION

The 8286 and 8287 transceivers are 8-bit transceivers with high impedance outputs. With T active HIGH and  $\overline{OE}$  active LOW, data at the A<sub>0</sub>-A<sub>7</sub> pins is driven onto the B<sub>0</sub>-B<sub>7</sub> pins. With T inactive LOW and  $\overline{OE}$  active LOW, data at the B<sub>0</sub>-B<sub>7</sub> pins is driven onto the A<sub>0</sub>-A<sub>7</sub> pins. No output low glitching will occur whenever the transceivers are entering or leaving the high impedance state.

## D.C. AND OPERATING CHARACTERISTICS

### ABSOLUTE MAXIMUM RATINGS\*

Temperature Under Bias . . . . . 0°C to 70°C  
Storage Temperature . . . . . - 65°C to + 150°C  
All Output and Supply Voltages . . . . . - 0.5V to + 7V  
All Input Voltages . . . . . - 1.0V to + 5.5V  
Power Dissipation . . . . . 1 Watt

\*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

## D.C. CHARACTERISTICS FOR 8286/8287

Conditions: V<sub>CC</sub> = 5V ± 5%, T<sub>A</sub> = 0°C to 70°C

Symbol	Parameter	Min	Max	Units	Test Conditions
V <sub>C</sub>	Input Clamp Voltage		-1	V	I <sub>C</sub> = -5 mA
I <sub>CC</sub>	Power Supply Current—8287 —8286		130 160	mA mA	
I <sub>F</sub>	Forward Input Current		-0.2	mA	V <sub>F</sub> = 0.45V
I <sub>R</sub>	Reverse Input Current		50	μA	V <sub>R</sub> = 5.25V
V <sub>OL</sub>	Output Low Voltage —B Outputs —A Outputs		0.5 0.5	V V	I <sub>OL</sub> = 32 mA I <sub>OL</sub> = 10 mA
V <sub>OH</sub>	Output High Voltage —B Outputs —A Outputs	2.4 2.4		V V	I <sub>OH</sub> = -5 mA I <sub>OH</sub> = -1 mA
I <sub>OFF</sub> I <sub>OFF</sub>	Output Off Current Output Off Current		I <sub>F</sub> I <sub>R</sub>		V <sub>OFF</sub> = 0.45V V <sub>OFF</sub> = 5.25V
V <sub>IL</sub>	Input Low Voltage —A Side —B Side		0.8 0.9	V V	V <sub>CC</sub> = 5.0V, See Note 1 V <sub>CC</sub> = 5.0V, See Note 1
V <sub>IH</sub>	Input High Voltage	2.0		V	V <sub>CC</sub> = 5.0V, See Note 1
C <sub>IN</sub>	Input Capacitance		12	pF	F = 1 MHz V <sub>BIAS</sub> = 2.5V, V <sub>CC</sub> = 5V T <sub>A</sub> = 25°C

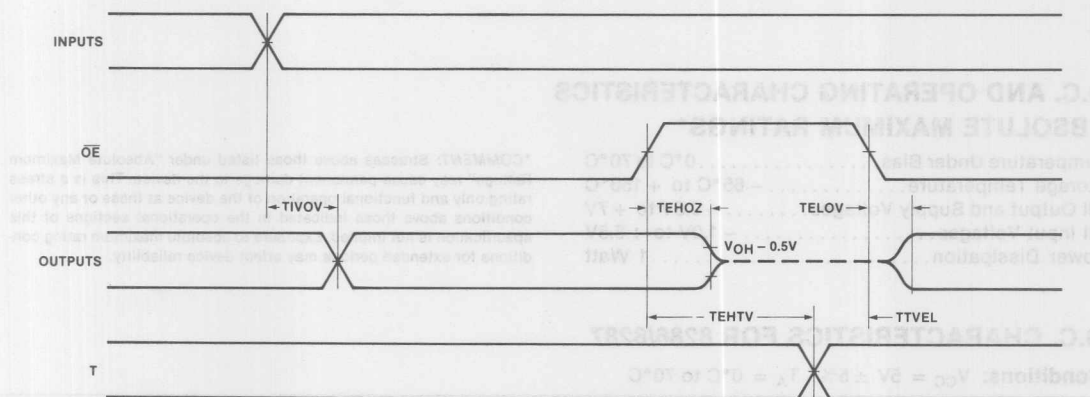
Note: 1. B Outputs — I<sub>OL</sub> = 32 mA, I<sub>OH</sub> = -5 mA, C<sub>L</sub> = 300 pF A Outputs — I<sub>OL</sub> = 10 mA, I<sub>OH</sub> = -1 mA, C<sub>L</sub> = 100 pF

**Loading:** B Outputs —  $I_{OL} = 32 \text{ mA}$ ,  $I_{OH} = -5 \text{ mA}$ ,  $C_L = 300 \text{ pF}$   
 A Outputs —  $I_{OL} = 10 \text{ mA}$ ,  $I_{OH} = -1 \text{ mA}$ ,  $C_L = 100 \text{ pF}$

Symbol	Parameter	Min	Max	Units	Test Conditions
TIVOV	Input to Output Delay				(See Note 1)
	Inverting		25	ns	
	Non-Inverting		35	ns	
TEHTV	Transmit/Receive Hold Time	TEHOZ		ns	
TTVEL	Transmit/Receive Setup	30		ns	
TEHOZ	Output Disable Time		25	ns	
TELOV	Output Enable Time	10	50	ns	

**Note:** 1. See waveforms and test load circuit on following page.

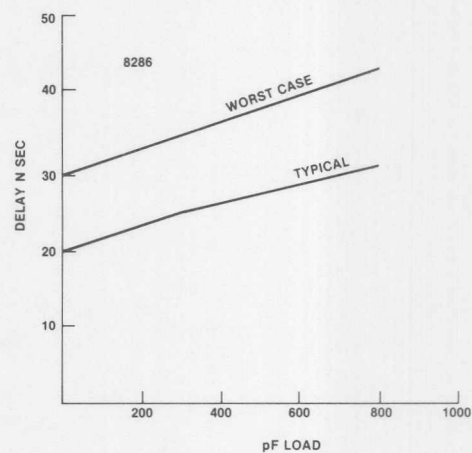
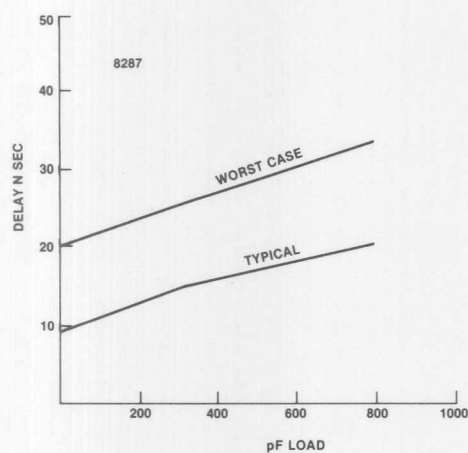
## 8286/8287 TIMING



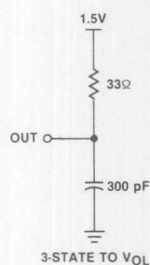
**NOTE:** 1. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS OTHERWISE NOTED.

Symbol	Parameter	Units	Min	Max	Test Conditions
$V_{CC}$	Input Clamp Voltage	V			
$I_{CC}$	Power Supply Current	mA		130	
$I_{CC}$	Power Supply Current	mA		180	
$I_F$	Forward Input Current	mA	-0.2		$V_F = 0.45 \text{ V}$
$I_R$	Reverse Input Current	$\mu\text{A}$	50		$V_R = 0.25 \text{ V}$
$V_{OL}$	Output Low Voltage	V	0.5		$I_{OL} = 32 \text{ mA}$
$V_{OL}$	Output Low Voltage	V	0.5		$I_{OL} = 10 \text{ mA}$
$V_{OH}$	Output High Voltage	V	2.4		$I_{OH} = -5 \text{ mA}$
$V_{OH}$	Output High Voltage	V	2.4		$I_{OH} = -1 \text{ mA}$
$I_{OH}$	Output Off Current	$\mu\text{A}$			$V_{OH} = 0.45 \text{ V}$
$I_{OH}$	Output Off Current	$\mu\text{A}$			$V_{OH} = 0.25 \text{ V}$
$V_{IL}$	Input Low Voltage	V	0.8		$V_{CC} = 0.5 \text{ V}$ , See Note 1
$V_{IL}$	Input Low Voltage	V	0.9		$V_{CC} = 2.0 \text{ V}$ , See Note 1
$V_{IH}$	Input High Voltage	V	2.0		$V_{CC} = 0.5 \text{ V}$ , See Note 1
$C_{IN}$	Input Capacitance	pF	12		$f = 1 \text{ MHz}$ $V_{OH} = 2.5 \text{ V}$ , $V_{OL} = 0.5 \text{ V}$ $T_A = 25^\circ\text{C}$

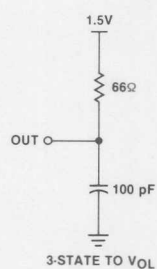
## OUTPUT DELAY VS. CAPACITANCE



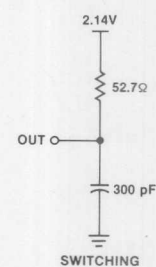
## TEST LOAD CIRCUITS



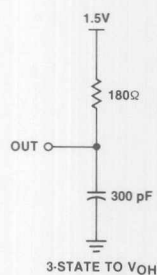
B OUTPUT



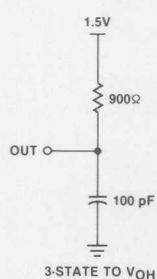
A OUTPUT



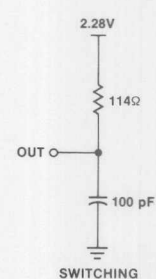
B OUTPUT



B OUTPUT



A OUTPUT



A OUTPUT





## Chapter 6

**MCS-85™**

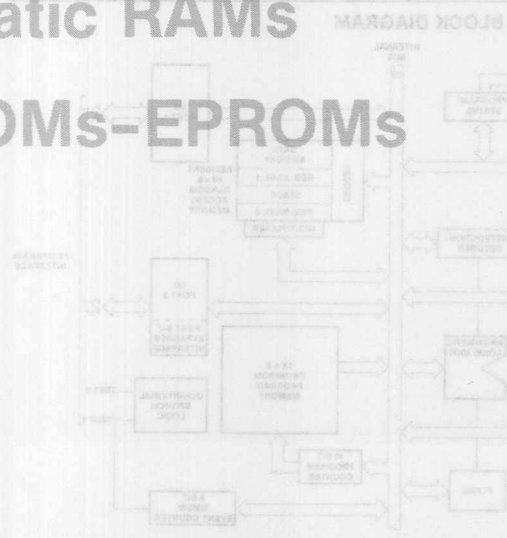
**MCS-80™**

## Systems Support Components

## Peripherals

## Static RAMs

## ROMs-EPROMs





## 8041A/8641A/8741A UNIVERSAL PERIPHERAL INTERFACE 8-BIT MICROCOMPUTER

- 8-Bit CPU plus ROM, RAM, I/O, Timer and Clock in a Single Package
- One 8-Bit Status and Two Data Registers for Asynchronous Slave-to-Master Interface
- DMA, Interrupt, or Polled Operation Supported
- 1024 × 8 ROM/EPROM, 64 × 8 RAM, 8-Bit Timer/Counter, 18 Programmable I/O Pins
- Fully Compatible with MCS-48™, MCS-80™, MCS-85™, and MCS-86™ Microprocessor Families
- Interchangeable ROM and EPROM Versions
- 3.6 MHz 8741A-8 Available
- Expandable I/O
- RAM Power-Down Capability
- Over 90 Instructions: 70% Single Byte
- Single 5V Supply

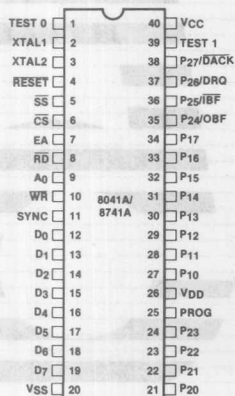
The Intel® 8041A/8741A is a general purpose, programmable interface device designed for use with a variety of 8-bit microprocessor systems. It contains a low cost microcomputer with program memory, data memory, 8-bit CPU, I/O ports, timer/counter, and clock in a single 40-pin package. Interface registers are included to enable the UPI device to function as a peripheral controller in MCS-48™, MCS-80™, MCS-85™, MCS-86™, and other 8-bit systems.

The UPI-41A™ has 1K words of program memory and 64 words of data memory on-chip. To allow full user flexibility the program memory is available as ROM in the 8041A version or as UV-erasable EPROM in the 8741A version. The 8741A and the 8041A are fully pin compatible for easy transition from prototype to production level designs. The 8641A is a one-time programmable (at the factory) 8741A which can be ordered as the first 25 pieces of a new 8041A order. The substitution of 8641A's for 8041A's allows for very fast turnaround for initial code verification and evaluation results.

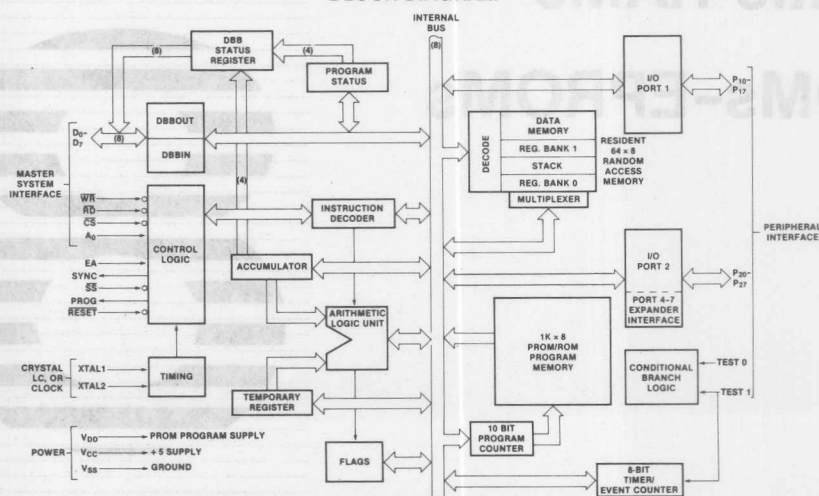
The device has two 8-bit, TTL compatible I/O ports and two test inputs. Individual port lines can function as either inputs or outputs under software control. I/O can be expanded with the 8243 device which is directly compatible and has 16 I/O lines. An 8-bit programmable timer/counter is included in the UPI device for generating timing sequences or counting external inputs. Additional UPI features include: single 5V supply, low power standby mode (in the 8041A), single-step mode for debug (in the 8741A), and dual working register banks.

Because it's a complete microcomputer, the UPI provides more flexibility for the designer than conventional LSI interface devices. It is designed to be an efficient controller as well as an arithmetic processor. Applications include keyboard scanning, printer control, display multiplexing and similar functions which involve interfacing peripheral devices to microprocessor systems.

### PIN CONFIGURATION



### BLOCK DIAGRAM

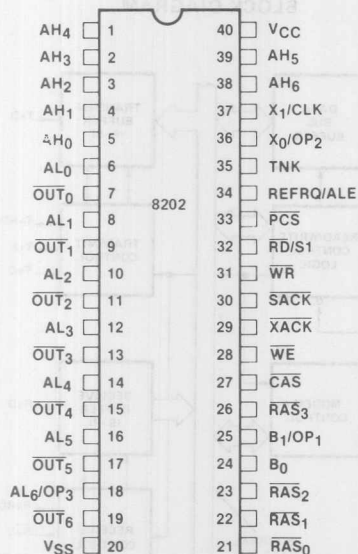


## 8202 DYNAMIC RAM CONTROLLER

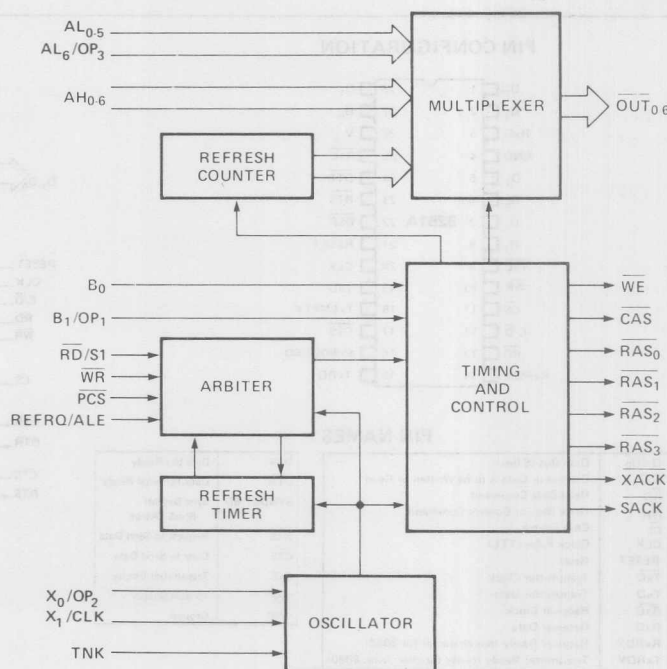
- Provides All Signals Necessary to Control 2104A, 2117, or 2118 Dynamic Memories
- Directly Addresses and Drives Up to 128K Bytes Without External Drivers
- Provides Address Multiplexing and Strobes
- Provides a Refresh Timer and a Refresh Counter
- Refresh Cycles May be Internally or Externally Requested
- Provides Transparent Refresh Capability
- Fully Compatible with Intel® 8080A, 8085A and 8086 Microprocessors
- Decodes 8085A Status for Advanced Read Capability
- Provides System Acknowledge and Transfer Acknowledge Signals
- Internal or External Clock Capability

The 8202 is a Dynamic RAM System Controller designed to provide all signals necessary to use 2104A, 2117, or 2118 Dynamic RAMs in microcomputer systems. The 8202 provides multiplexed addresses and address strobes, as well as refresh/access arbitration. Refresh cycles can be started internally or externally.

PIN CONFIGURATION



8202 BLOCK DIAGRAM

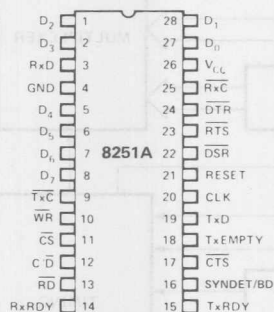


# PROGRAMMABLE COMMUNICATION INTERFACE

- Synchronous and Asynchronous Operation
- Synchronous 5-8 Bit Characters; Internal or External Character Synchronization; Automatic Sync Insertion
- Asynchronous 5-8 Bit Characters; Clock Rate—1, 16 or 64 Times Baud Rate; Break Character Generation; 1, 1½, or 2 Stop Bits; False Start Bit Detection; Automatic Break Detect and Handling.
- Synchronous Baud Rate — DC to 64K Baud
- Asynchronous Baud Rate — DC to 19.2K Baud
- Full Duplex, Double Buffered, Transmitter and Receiver
- Error Detection — Parity, Overrun and Framing
- Fully Compatible with 8080/8085 CPU
- 28-Pin DIP Package
- All Inputs and Outputs are TTL Compatible
- Single +5V Supply
- Single TTL Clock

The Intel® 8251A is the enhanced version of the industry standard, Intel® 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel's new high performance family of microprocessors such as the 8085. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is constructed using N-channel silicon gate technology.

## PIN CONFIGURATION

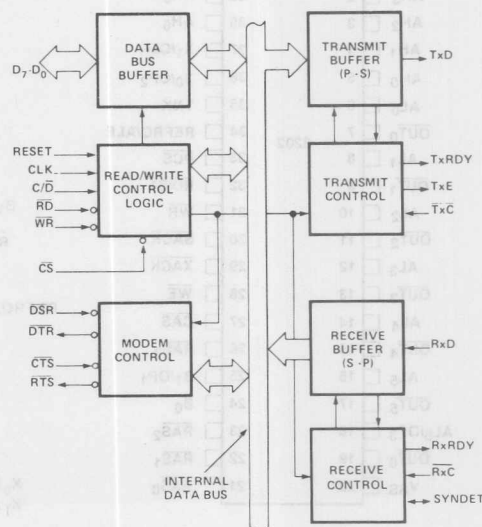


## PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
RD	Read Data Command
WR	Write Data or Control Command
CS	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
TxC	Transmitter Clock
TxD	Transmitter Data
RxC	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char from 8080)

DSR	Data Set Ready
DTR	Data Terminal Ready
SYNDET/BD	Sync Detect/ Break Detect
RTS	Request to Send Data
CTS	Clear to Send Data
TxE	Transmitter Empty
V <sub>CC</sub>	+5 Volt Supply
GND	Ground

## BLOCK DIAGRAM



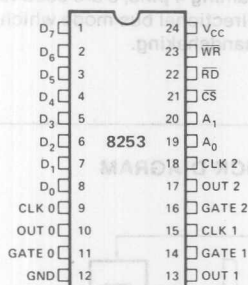
# 8253/8253-5 PROGRAMMABLE INTERVAL TIMER

- MCS-85™ Compatible 8253-5
- Count Binary or BCD
- 3 Independent 16-Bit Counters
- Single +5V Supply
- DC to 2 MHz
- Programmable Counter Modes
- 24-Pin Dual In-Line Package

The Intel® 8253 is a programmable counter/timer chip designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2 MHz. All modes of operation are software programmable.

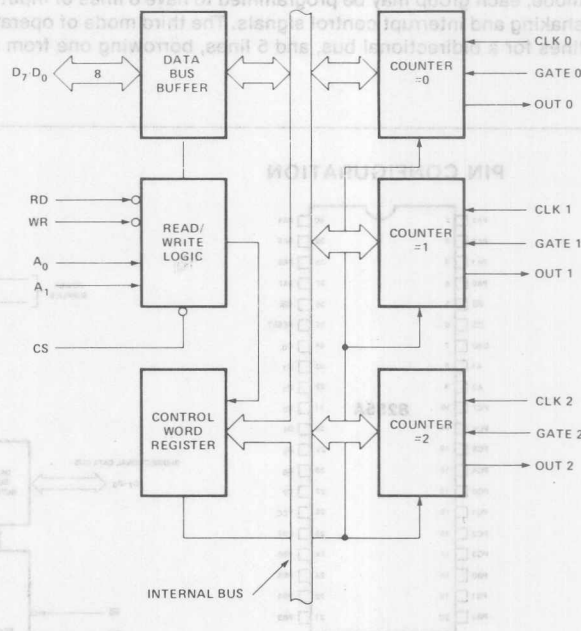
## PIN CONFIGURATION



## PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	DATA BUS (8-BIT)
CLK N	COUNTER CLOCK INPUTS
GATE N	COUNTER GATE INPUTS
OUT N	COUNTER OUTPUTS
RD	READ COUNTER
WR	WRITE COMMAND OR DATA
CS	CHIP SELECT
A <sub>0</sub> , A <sub>1</sub>	COUNTER SELECT
V <sub>CC</sub>	+5 VOLTS
GND	GROUND

## BLOCK DIAGRAM



## PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	DATA BUS (8-BIT)
RESET	RESET INPUT
CS	CHIP SELECT
RD	READ COUNTER
WR	WRITE COMMAND OR DATA
A <sub>0</sub> , A <sub>1</sub>	COUNTER SELECT
V <sub>CC</sub>	+5 VOLTS
GND	GROUND

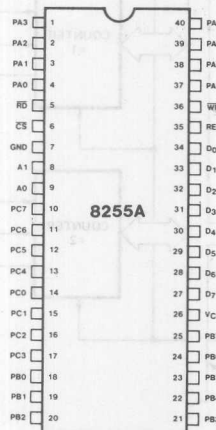


## 8255A/8255A-5 PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel® Micro-processor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- 40-Pin Dual In-Line Package
- Reduces System Package Count
- Improved DC Driving Capability

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel® microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

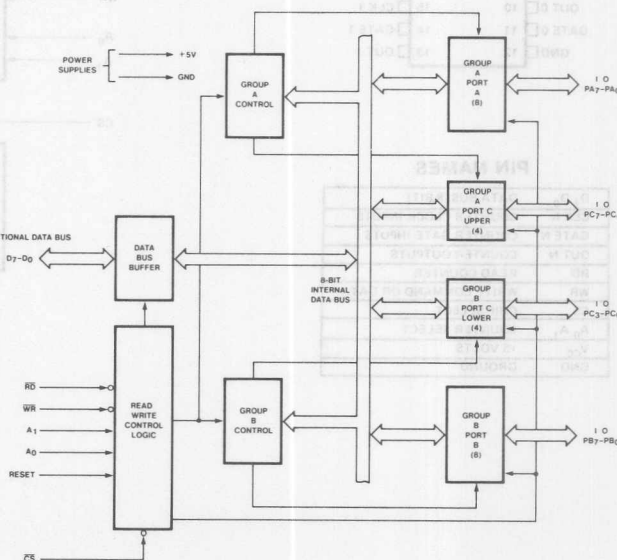
PIN CONFIGURATION



PIN NAMES

D <sub>7</sub> -D <sub>0</sub>	DATA BUS (BI DIRECTIONAL)
RESET	RESET INPUT
CS	CHIP SELECT
RD	READ INPUT
WR	WRITE INPUT
A <sub>0</sub> , A <sub>1</sub>	PORT ADDRESS
PA <sub>7</sub> -PA <sub>0</sub>	PORT A (BIT)
PB <sub>7</sub> -PB <sub>0</sub>	PORT B (BIT)
PC <sub>7</sub> -PC <sub>0</sub>	PORT C (BIT)
V <sub>CC</sub>	+5 VOLTS
GND	0 VOLTS

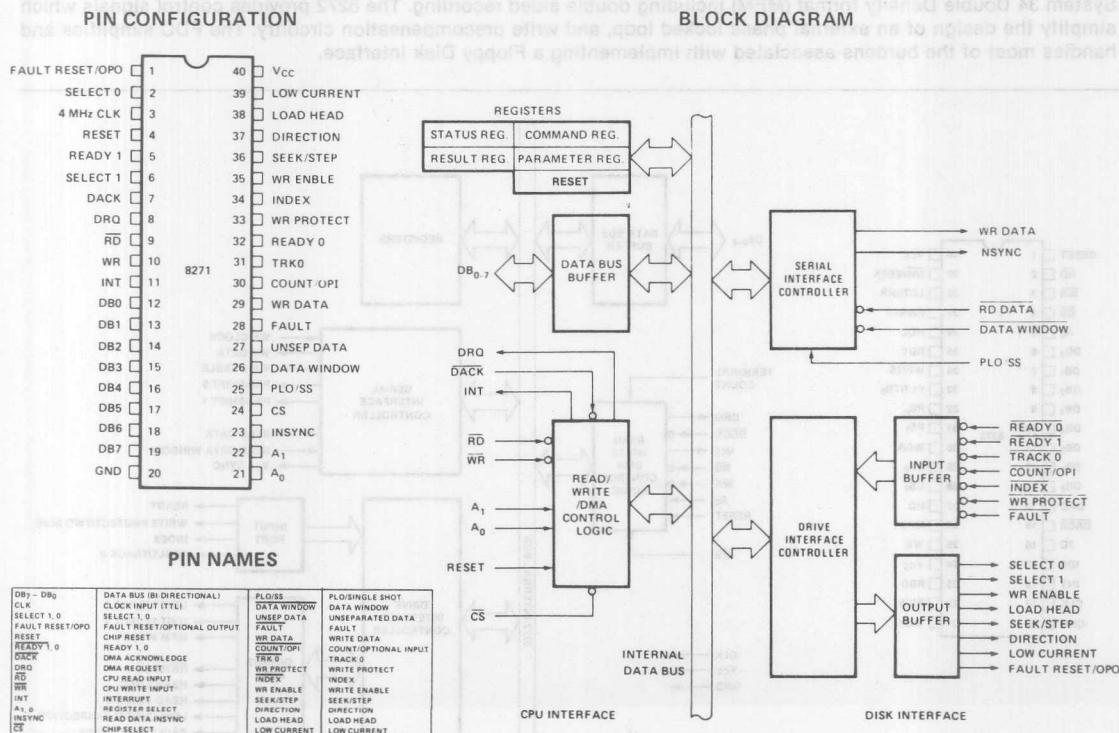
8255A BLOCK DIAGRAM



# PROGRAMMABLE FLOPPY DISK CONTROLLER

- IBM 3740 Soft Sectored Format Compatible
- Programmable Record Lengths
- Multi-Sector Capability
- Maintain Dual Drives with Minimum Software Overhead Expandable to 4 Drives
- Automatic Read/Write Head Positioning and Verification
- Internal CRC Generation and Checking
- Programmable Step Rate, Settle-Time, Head Load Time, Head Unload Index Count
- Fully MCS-80™ and MCS-85™ Compatible
- Single +5V Supply
- 40-Pin Package

The Intel® 8271 Programmable Floppy Disk Controller (FDC) is an LSI component designed to interface one to 4 floppy disk drives to an 8-bit microcomputer system. Its powerful control functions minimize both hardware and software overhead normally associated with floppy disk controllers.

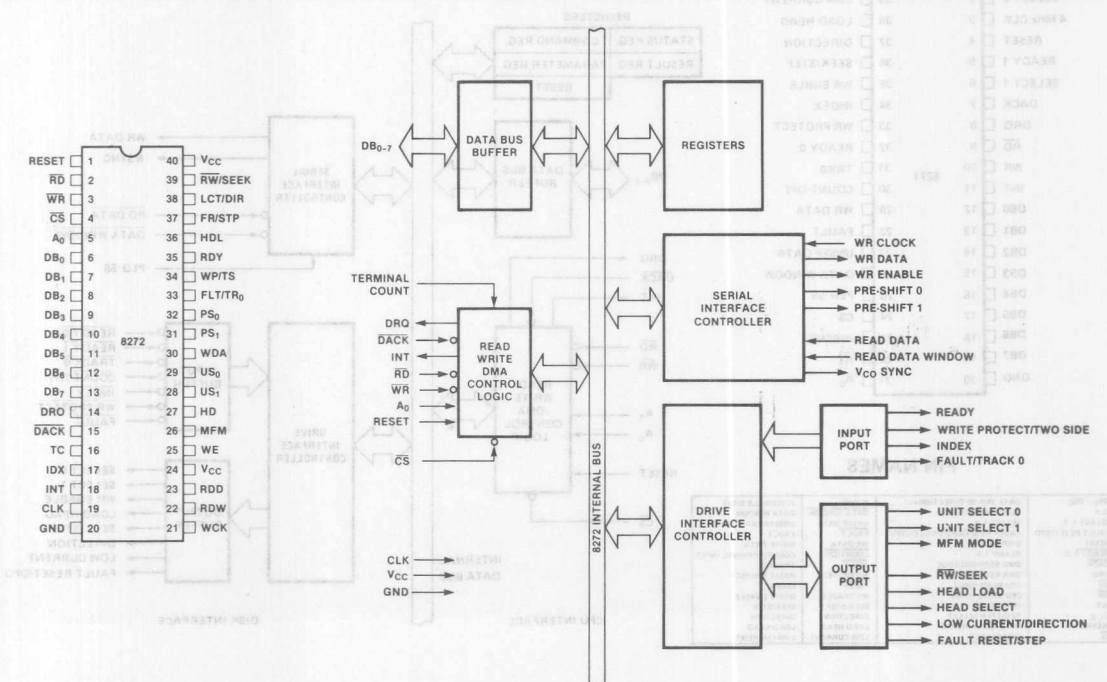


# SINGLE/DOUBLE DENSITY FLOPPY DISK CONTROLLER

**ADVANCE INFORMATION**  
Characteristics are subject to change without notice.

- IBM Compatible in Both Single and Double Density Recording Formats
- Programmable Data Record Lengths: 128, 256, 512, or 1024 Bytes/Sector
- Multi-Sector and Multi-Track Transfer Capability
- Drive Up to 4 Floppy Disks
- Data Scan Capability — Will Scan a Single Sector or an Entire Cylinder's Worth of Data Fields, Comparing on a Byte by Byte Basis, Data in the Processor's Memory with Data Read from the Diskette
- Data Transfers in DMA or Non-DMA Mode
- Parallel Seek Operations on Up to Four Drives
- Compatible with Most Microprocessors Including 8080A, 8085A, and 8086
- Single Phase 8 MHz Clock
- Single +5 Volt Power Supply
- Available in 40-Pin Plastic and CERPID Dual-in-Line Package

The 8272 is an LSI Floppy Disk Controller (FDC) Chip, which contains the circuitry and control functions for interfacing a processor to 4 Floppy Disk Drives. It is capable of supporting either IBM 3740 single density format (FM), or IBM System 34 Double Density format (MFM) including double sided recording. The 8272 provides control signals which simplify the design of an external phase locked loop, and write precompensation circuitry. The FDC simplifies and handles most of the burdens associated with implementing a Floppy Disk Interface.

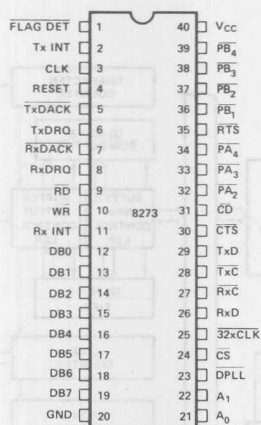


# 8273 PROGRAMMABLE HDLC/SDLC PROTOCOL CONTROLLER

- HDLC/SDLC Compatible
- Frame Level Commands
- Full Duplex, Half Duplex, or Loop SDLC Operation
- Up to 64K Baud Transfers
- Two User Programmable Modem Control Ports
- Automatic FCS (CRC) Generation and Checking
- Programmable NRZI Encode/Decode
- N-Bit Reception Capability
- Digital Phase Locked Loop Clock Recovery
- Minimum CPU Overhead
- Fully Compatible with 8080/8085 CPUss
- Single +5V Supply
- 40-Pin Package

The Intel® 8273 Programmable HDLC/SDLC Protocol Controller is a dedicated device designed to support the ISO/CITT's HDLC and IBM's SDLC communication line protocols. It is fully compatible with Intel's new high performance microcomputer systems such as the MCS-85™. A frame level command set is achieved by a unique microprogrammed dual processor chip architecture. The processing capability supported by the 8273 relieves the system CPU of the low level real-time tasks normally associated with controllers.

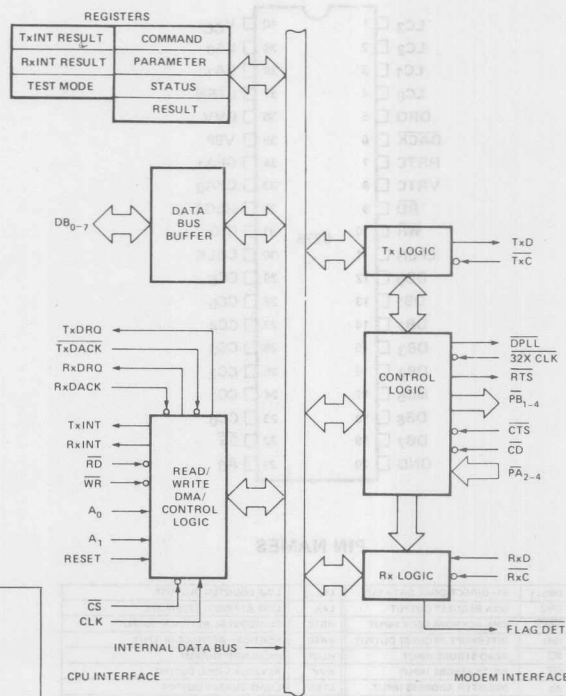
PIN CONFIGURATION



PIN NAMES

DB0-DB7	DATA BUS (8 BITS)	CS	CHIP SELECT
FLAG DET	FLAG DETECT	32XCLK	32 TIMES CLOCK
TxINT	TRANSMITTER INTERRUPT	RxD	RECEIVER DATA
CLK	CLOCK INPUT	RxC	RECEIVER CLOCK
RESET	RESET	TxC	TRANSMITTER DATA
TxDACK	TRANSMITTER DMA ACKNOWLEDGE	TxD	TRANSMITTER DATA
TxDREQ	TRANSMITTER DMA REQUEST	CTS	CLEAR TO SEND
RD	READ INPUT	CD	CARRIER DETECT
WR	WRITE INPUT	PA2-PA4	GP INPUT PORTS
RxDACK	RECEIVER DMA ACKNOWLEDGE	PB1-PB4	GP OUTPUT PORTS
RxDREQ	RECEIVER DMA REQUEST	RTS	REQUEST TO SEND
RxINT	RECEIVER INTERRUPT	VCC	+5 VOLT SUPPLY
A0-A1	COMMAND REGISTER SELECT ADDRESS	GND	GROUND
DPLL	DIGITAL PHASE LOCKED LOOP		

BLOCK DIAGRAM



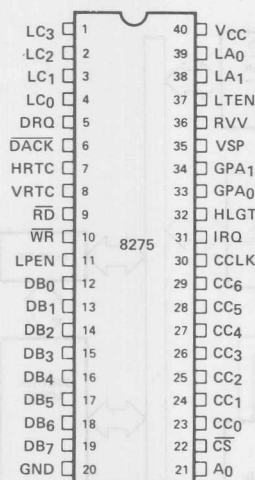
# 8275

## PROGRAMMABLE CRT CONTROLLER

- Programmable Screen and Character Format
- 6 Independent Visual Field Attributes
- 11 Visual Character Attributes (Graphic Capability)
- Cursor Control (4 Types)
- Light Pen Detection and Registers
- Fully MCS-80™ and MCS-85™ Compatible
- Dual Row Buffers
- Programmable DMA Burst Mode
- Single +5V Supply
- 40-Pin Package

The Intel® 8275 Programmable CRT Controller is a single chip device to interface CRT raster scan displays with Intel® microcomputer systems. Its primary function is to refresh the display by buffering the information from main memory and keeping track of the display position of the screen. The flexibility designed into the 8275 will allow simple interface to almost any raster scan CRT display with a minimum of external hardware and software overhead.

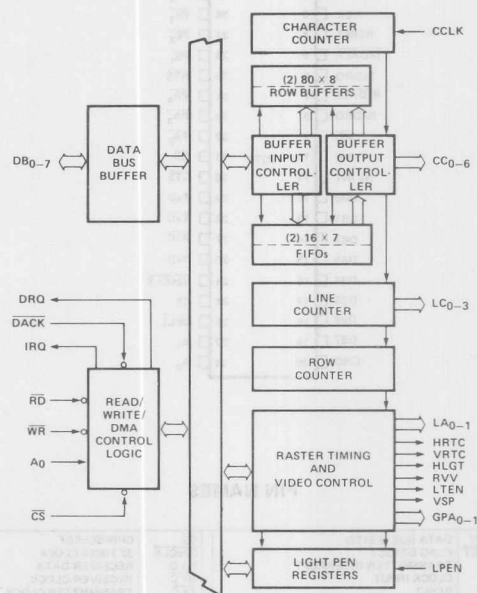
### PIN CONFIGURATION



### PIN NAMES

DB <sub>0-7</sub>	B <sub>1</sub> -DIRECTIONAL DATA BUS	LC <sub>0-3</sub>	LINE COUNTER OUTPUTS
DRQ	DMA REQUEST OUTPUT	LA <sub>0-1</sub>	LINE ATTRIBUTE OUTPUTS
DACK	DMA ACKNOWLEDGE INPUT	HRTC	HORIZONTAL RETRACE OUTPUT
IRQ	INTERRUPT REQUEST OUTPUT	VRTC	VERTICAL RETRACE OUTPUT
RD	READ STROBE INPUT	HLGT	HIGHLIGHT OUTPUT
WR	WRITE STROBE INPUT	RVV	REVERSE VIDEO OUTPUT
A <sub>0</sub>	REGISTER ADDRESS INPUT	LTEN	LIGHT ENABLE OUTPUT
CS	CHIP SELECT INPUT	VSP	VIDEO SUPPRESS OUTPUT
CCLK	CHARACTER CLOCK INPUT	GPA <sub>0-1</sub>	GENERAL PURPOSE ATTRIBUTE OUTPUTS
CC <sub>0-6</sub>	CHARACTER CODE OUTPUTS	LPEN	LIGHT PEN INPUT

### BLOCK DIAGRAM





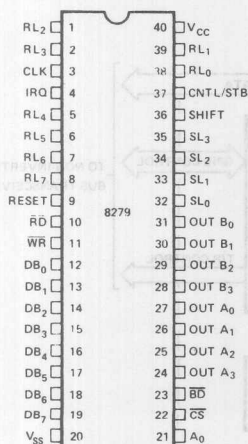
## 8279/8279-5 PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

- MCS-85™ Compatible 8279-5
- Simultaneous Keyboard Display Operations
- Scanned Keyboard Mode
- Scanned Sensor Mode
- Strobed Input Entry Mode
- 8-Character Keyboard FIFO
- 2-Key Lockout or N-Key Rollover with Contact Debounce
- Dual 8- or 16-Numerical Display
- Single 16-Character Display
- Right or Left Entry 16-Byte Display RAM
- Mode Programmable from CPU
- Programmable Scan Timing
- Interrupt Output on Key Entry

The Intel® 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel® microprocessors. The keyboard portion can provide a scanned interface to a 64-contact key matrix. The keyboard portion will also interface to an array of sensors or a strobed interface keyboard, such as the hall effect and ferrite variety. Key depressions can be 2-key lockout or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO. If more than 8 characters are entered, overrun status is set. Key entries set the interrupt output line to the CPU.

The display portion provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used as well as simple indicators. The 8279 has 16X8 display RAM which can be organized into dual 16X4. The RAM can be loaded or interrogated by the CPU. Both right entry, calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

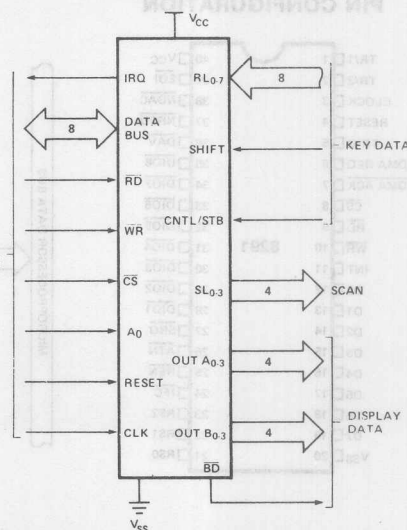
### PIN CONFIGURATION



### PIN NAMES

DB <sub>0-7</sub>	I/O	DATA BUS (BI DIRECTIONAL)
CLK	I	CLOCK INPUT
RESET	I	RESET INPUT
CS	I	CHIP SELECT
RD	I	READ INPUT
WR	I	WRITE INPUT
A <sub>0</sub>	I	BUFFER ADDRESS
IRQ	O	INTERRUPT REQUEST OUTPUT
SL <sub>0-3</sub>	O	SCAN LINES
RL <sub>0-7</sub>	I	RETURN LINES
SHIFT	I	SHIFT INPUT
CNTL/STB	I	CONTROL/STROBE INPUT
OUT A <sub>0-3</sub>	O	DISPLAY (A) OUTPUTS
OUT B <sub>0-3</sub>	O	DISPLAY (B) OUTPUTS
BD	O	BLANK DISPLAY OUTPUT

### LOGIC SYMBOL

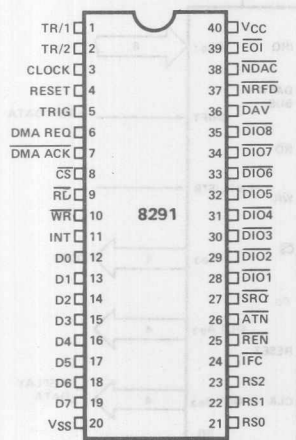


# **GPIB TALKER/LISTENER**

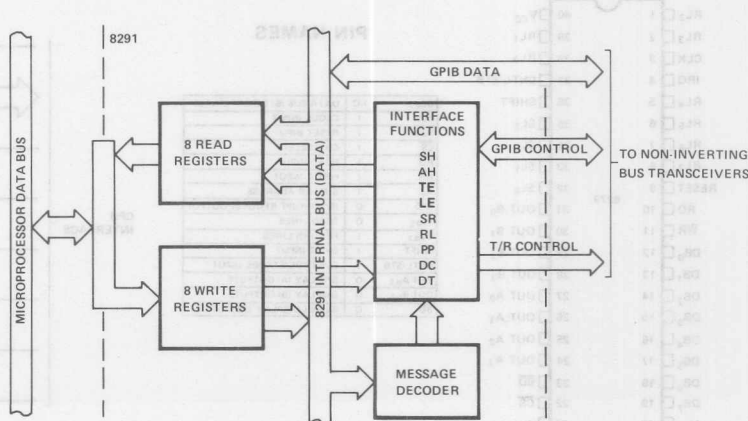
- Designed to Interface Microprocessors (e.g., 8080, 8085, 8086, 8048) to an IEEE Standard 488 Digital Interface Bus
- Programmable Data Transfer Rate
- Complete Source and Acceptor Handshake
- Complete Talker and Listener Functions with Extended Addressing
- Service Request, Parallel Poll, Device Clear, Device Trigger, Remote/Local Functions
- Selectable Interrupts
- On-Chip Primary and Secondary Address Recognition
- Automatic Handling of Addressing and Handshake Protocol
- Provision for Software Implementation of Additional Features
- 1 – 8 MHz Clock Range
- 16 Registers (8 Read, 8 Write), 2 for Data Transfer, the Rest for Interface Function Control, Status, etc.
- Directly Interfaces to External Non-Inverting Transceivers for Connection to the GPIB
- Provides Three Addressing Modes, Allowing the Chip to be Addressed Either as a Major or a Minor Talker/Listener with Primary or Secondary Addressing
- DMA Handshake Provision Allows for Bus Transfers without CPU Intervention
- Trigger Output Pin
- On-Chip EOS (End of Sequence) Message Recognition Facilitates Handling of Multi-Byte Transfers

The 8291 GPIB Talker/Listener is a microprocessor-controlled chip designed to interface microprocessors (e.g., 8048, 8080, 8085, 8086) to an IEEE Standard 488 Instrumentation Interface Bus. It implements all of the Standard's interface functions except for the controller.

## **PIN CONFIGURATION**



## **BLOCK DIAGRAM**



# 8292 GPIB CONTROLLER

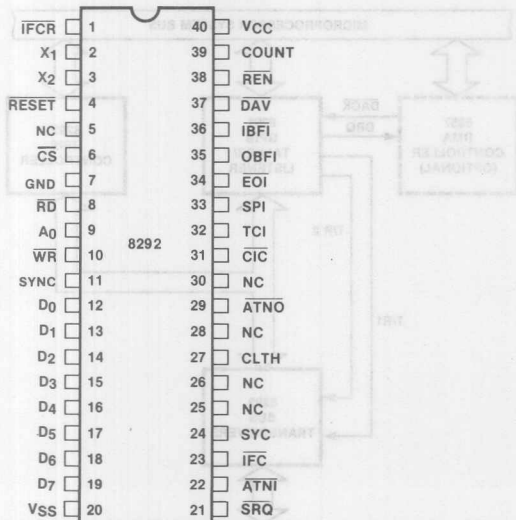
**PRELIMINARY**  
Notice: This is not a final specification. Some parametric limits are subject to change.

## FEATURES:

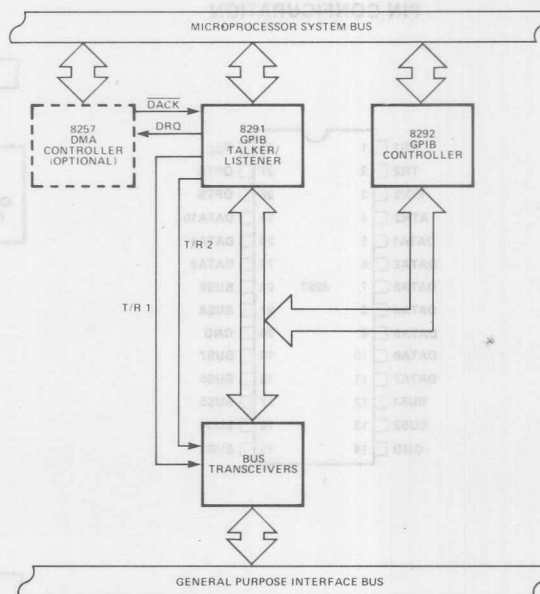
- Complete IEEE Standard 488 Controller Function.
- Interface Clear (IFC) Sending Capability Allows for Seizure of Control and/or Initialization of the Bus.
- Responds to Service Requests (SRQ).
- Sends (REN), Allowing Instruments to Switch to Remote Control.
- Complete Implementation of Transfer Control Protocol.
- Synchronous Control Seizure Prevents the Destruction of any Data Transmission in Progress.
- Connects with the 8291 to Form a Complete IEEE Standard 488 Interface Talker/Listener/Controller.

The 8292 GPIB CONTROLLER is a microprocessor-controlled chip designed to connect with the 8291 GPIB TALKER/LISTENER to implement the full IEEE Standard 488 controller function, including transfer control protocol. The 8292 is a pre-programmed UPI-41A™

## PIN CONFIGURATION



## 8291, 8292 SYSTEM DIAGRAM

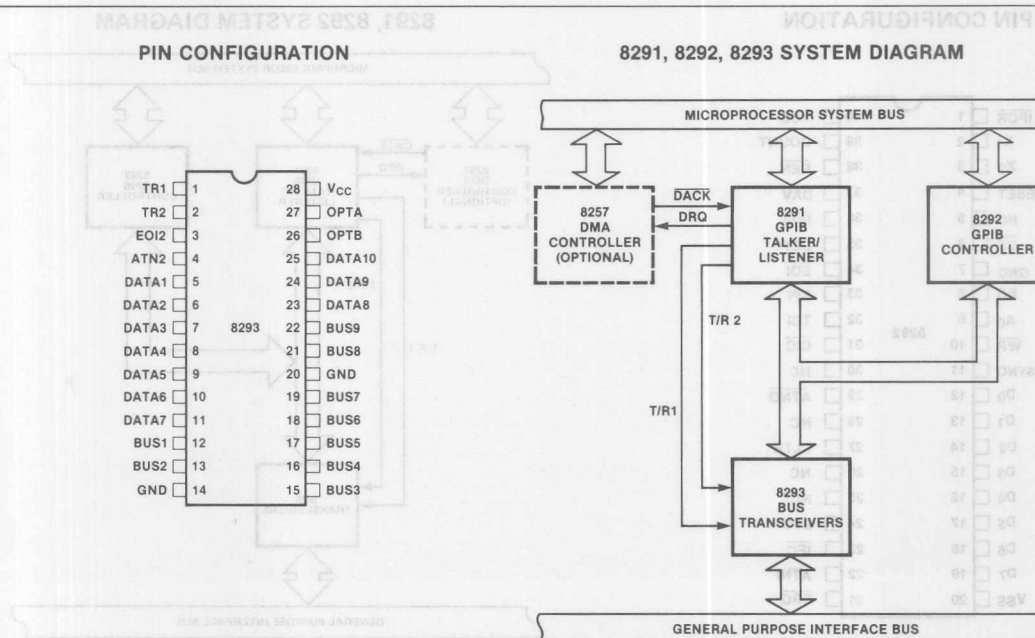


# 8293 GPIO TRANSCEIVER

**ADVANCE INFORMATION**  
Characteristics are subject to change without notice.

- Nine Open-collector or Three-state Line Drivers
- 48 mA Sink Current Capability on Each Line Driver
- Nine Schmitt-type Line Receivers
- High Capacitance Load Drive Capability
- Single 5V Power Supply
- 28-Pin Package
- Low Power HMOS Design
- On-chip Decoder for Mode Configuration
- Power Up/Power Down Protection to Prevent Disrupting the IEEE Bus
- Connects with the 8291 and 8292 to Form an IEEE Standard 488 Interface Talker/Listener/Controller with no Additional Components
- Only Two 8293's Required per GPIO Interface

The Intel® 8293 GPIO Transceiver is a high current, non-inverting buffer chip designed to interface the 8291 GPIO Talker/Listener or the 8292 GPIO Controller with the 8291 to the IEEE Standard 488-1978 Instrumentation Interface Bus. Each GPIO interface would contain two 8293 Bus Transceivers. In addition, the 8293 can also be used as a general purpose bus driver.



# DATA ENCRYPTION UNIT

**PRELIMINARY**  
Notice: This is not a final specification. Some parametric limits are subject to change.

- Certified by National Bureau of Standards
- 80 Byte/Sec Data Conversion Rate
- 64-Bit Data Encryption Using 56-Bit Key
- DMA Interface
- 3 Interrupt Outputs to Aid in Loading and Unloading Data
- 7-Bit User Output Port
- Single 5V  $\pm$  10% Power Supply
- Peripheral to MCS-86™, MCS-85™, MCS-80™ and MCS-48™ Processors
- Implements Federal Information Processing Data Encryption Standard
- Encrypt and Decrypt Modes Available

## DESCRIPTION

The Intel® 8294 Data Encryption Unit (DEU) is a microprocessor peripheral device designed to encrypt and decrypt 64-bit blocks of data using the algorithm specified in the Federal Information Processing Data Encryption Standard. The DEU operates on 64-bit text words using a 56-bit user-specified key to produce 64-bit cipher words. The operation is reversible: if the cipher word is operated upon, the original text word is produced. The algorithm itself is permanently contained in the 8294; however, the 56-bit key is user-defined and may be changed at any time.

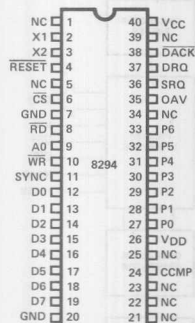
The 56-bit key and 64-bit message data are transferred to and from the 8294 in 8-bit bytes by way of the system data bus. A DMA interface and three interrupt outputs are available to minimize software overhead associated with data transfer. Also, by using the DMA interface two or more DEUs may be operated in parallel to achieve effective system conversion rates which are virtually any multiple of 80 bytes/second. The 8294 also has a 7-bit TTL compatible output port for user-specified functions.

Because the 8294 implements the NBS encryption algorithm it can be used in a variety of Electronic Funds Transfer applications as well as other electronic banking and data handling applications where data must be encrypted.

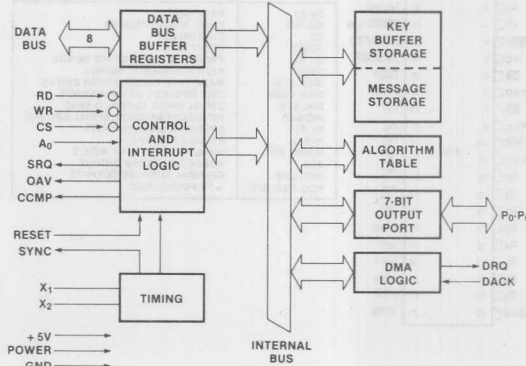
## PIN CONFIGURATION

## PIN NAMES

## BLOCK DIAGRAM



PIN NAME	FUNCTION
D <sub>7</sub> -D <sub>0</sub>	DATA BUS
RD, WR	READ, WRITE STROBES
CS	CHIP SELECT
A <sub>0</sub>	CONTROL/DATA SELECT
RESET	RESET INPUT
X <sub>1</sub> , X <sub>2</sub>	FREQUENCY REFERENCE INPUT
SYNC	HIGH FREQUENCY OUTPUT
DMA, DACK	DMA REQUEST, DMA ACKNOWLEDGE
SRQ, OAV, CCMP	INTERRUPT REQUEST OUTPUTS
P <sub>6</sub> -P <sub>0</sub>	OUTPUT PORT LINES
V <sub>CC</sub> , V <sub>DD</sub> , GND	+ 5V POWER, GND
NC	NO CONNECTION





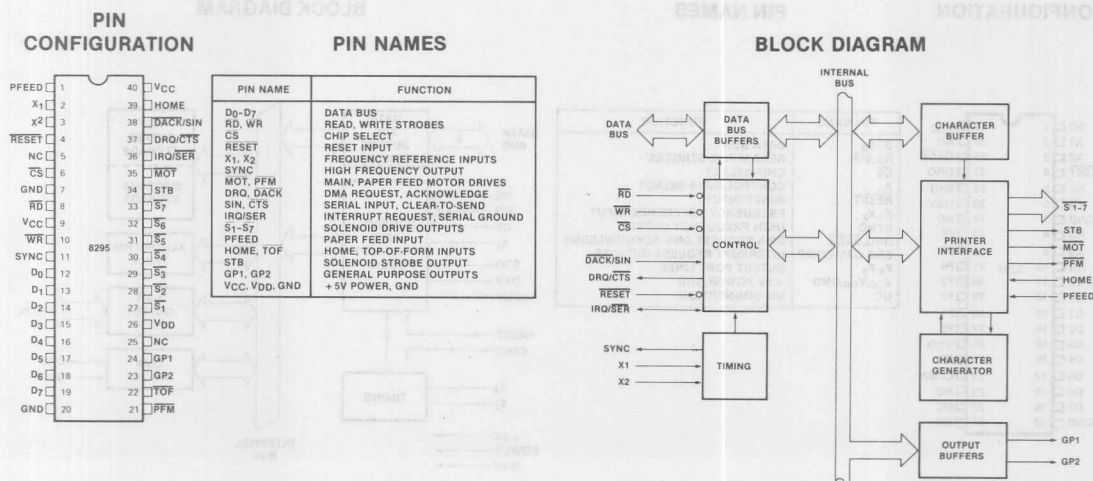
# 8295 DOT MATRIX PRINTER CONTROLLER

**PRELIMINARY**  
Notice: This is not a final specification. Some parametric limits are subject to change.

- Interfaces Dot Matrix Printers to MCS-48™, MCS-80/85™, MCS-86™ Systems
- 40 Character Buffer On Chip
- Serial or Parallel Communication with Host
- DMA Transfer Capability
- Programmable Character Density (10 or 12 Characters/Inch)
- Programmable Print Intensity
- Single or Double Width Printing
- Programmable Multiple Line Feeds
- 3 Tabulations
- 2 General Purpose Outputs

The Intel® 8295 Dot Matrix Printer Controller provides an interface for microprocessors to the LRC 7040 Series dot matrix impact printers. It may also be used as an interface to other similar printers.

The chip may be used in a serial or parallel communication mode with the host processor. In parallel mode, data transfers are based on polling, interrupts, or DMA. Furthermore, it provides internal buffering of up to 40 characters and contains a 7 × 7 matrix character generator accommodating 64 ASCII characters.



## Chapter 6

MCS-85™

MCS-80™

Systems  
Support  
Components

Peripherals

Static RAMs

ROMs-EPROMs

MOS

60

MOS

60

Chapter 6  
MCS-85  
MCS-80  
Systems  
Support  
Components  
Peripherals  
Static RAMs  
ROMs-EPROMs



## 2114A

### 1024 X 4 BIT STATIC RAM

	2114AL-2	2114AL-3	2114AL-4	2114A-4	2114A-5
Max. Access Time (ns)	120	150	200	200	250
Max. Current (mA)	40	40	40	70	70

- HMOS Technology
- Low Power, High Speed
- Identical Cycle and Access Times
- Single +5V Supply  $\pm 10\%$
- High Density 18 Pin Package
- Completely Static Memory - No Clock or Timing Strobe Required
- Directly TTL Compatible: All Inputs and Outputs
- Common Data Input and Output Using Three-State Outputs
- 2114 Replacement

The Intel 2114A is a 4096-bit static Random Access Memory organized as 1024 words by 4-bits using HMOS, a high performance MOS technology. It uses fully DC stable (static) circuitry throughout, in both the array and the decoding, therefore it requires no clocks or refreshing to operate. Data access is particularly simple since address setup times are not required. The data is read out nondestructively and has the same polarity as the input data. Common input/output pins are provided.

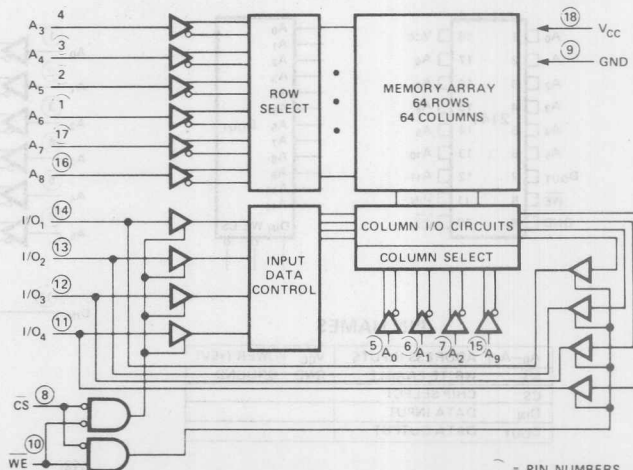
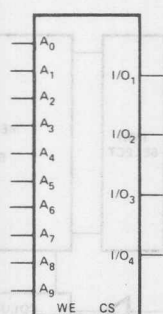
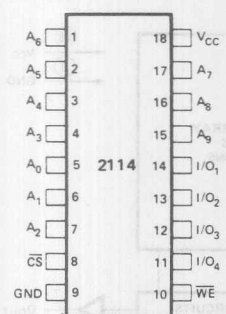
The 2114A is designed for memory applications where the high performance and high reliability of HMOS, low cost, large bit storage, and simple interfacing are important design objectives. The 2114A is placed in an 18-pin package for the highest possible density.

It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. A separate Chip Select ( $\overline{CS}$ ) lead allows easy selection of an individual package when outputs are or-tied.

#### PIN CONFIGURATION

#### LOGIC SYMBOL

#### BLOCK DIAGRAM



#### PIN NAMES

$A_0-A_9$	ADDRESS INPUTS	$V_{CC}$	POWER (+5V)
WE	WRITE ENABLE	GND	GROUND
CS	CHIP SELECT		
$I/O_1-I/O_4$	DATA INPUT/OUTPUT		

# 2141 4096 X 1 BIT STATIC RAM

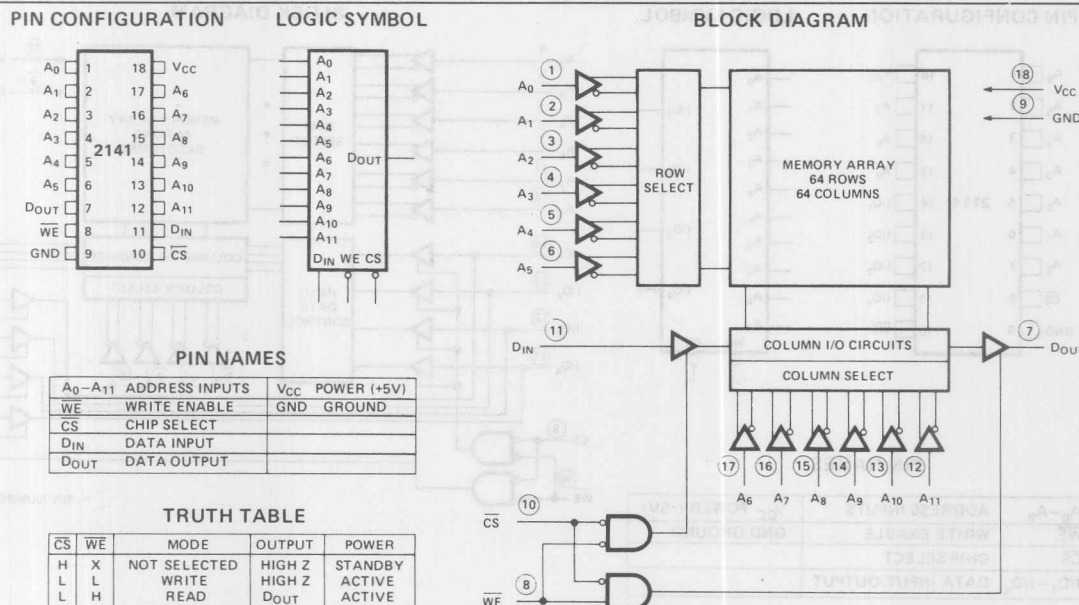
	2141-2	2141-3	2141-4	2141-5	2141L-3	2141L-4	2141L-5
Max. Access Time (ns)	120	150	200	250	150	200	250
Max. Active Current (mA)	70	70	55	55	40	40	40
Max. Standby Current (mA)	20	20	12	12	5	5	5

- HMOS Technology
- Industry Standard 2147 Pinout
- Completely Static Memory — No Clock or Timing Strobe Required
- Equal Access and Cycle Times
- Single +5V Supply
- Automatic Power-Down
- Directly TTL Compatible — All Inputs and Output
- Separate Data Input and Output
- Three-State Output
- High Density 18-Pin Package

The Intel® 2141 is a 4096-bit static Random Access Memory organized as 4096 words by 1-bit using HMOS, a high-performance MOS technology. It uses a uniquely innovative design approach which provides the ease-of-use features associated with non-clocked static memories and the reduced standby power dissipation associated with clocked static memories. To the user this means low standby power dissipation without the need for clocks, address setup and hold times, nor reduced data rates due to cycle times that are longer than access times.

$\overline{CS}$  controls the power-down feature. In less than a cycle time after  $\overline{CS}$  goes high — deselecting the 2141 — the part automatically reduces its power requirements and remains in this low power standby mode as long as  $\overline{CS}$  remains high. This device feature results in system power savings as great as 85% in larger systems, where the majority of devices are deselected.

The 2141 is placed in an 18-pin package configured with the industry standard pinout, the same as the 2147. It is directly TTL compatible in all respects: inputs, output, and a single +5V supply. The data is read out nondestructively and has the same polarity as the input data. A data input and a separate three-state output are used.





# 2142

## 1024 X 4 BIT STATIC RAM

	2142-2	2142-3	2142	2142L2	2142L3	2142L
Max. Access Time (ns)	200	300	450	200	300	450
Max. Power Dissipation (mw)	525	525	525	370	370	370

- High Density 20 Pin Package
- Access Time Selections From 200-450ns
- Identical Cycle and Access Times
- Low Operating Power Dissipation  
.1mW/Bit Typical
- Single +5V Supply
- No Clock or Timing Strobe Required
- Completely Static Memory
- Directly TTL Compatible: All Inputs and Outputs
- Common Data Input and Output Using Three-State Outputs

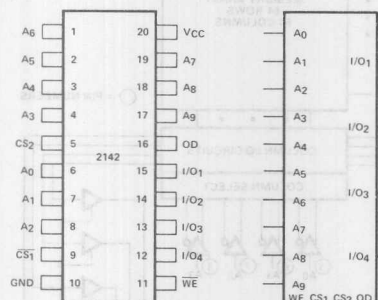
The Intel® 2142 is a 4096-bit static Random Access Memory organized as 1024 words by 4-bits using N-channel Silicon-Gate MOS technology. It uses fully DC stable (static) circuitry throughout — in both the array and the decoding — and therefore requires no clocks or refreshing to operate. Data access is particularly simple since address setup times are not required. The data is read out nondestructively and has the same polarity as the input data. Common input/output pins are provided.

The 2142 is designed for memory applications where high performance, low cost, large bit storage, and simple interfacing are important design objectives. It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply.

The 2142 is placed in a 20-pin package. Two Chip Selects (CS<sub>1</sub> and CS<sub>2</sub>) are provided for easy and flexible selection of individual packages when outputs are OR-tied. An Output Disable is included for direct control of the output buffers.

The 2142 is fabricated with Intel's N-channel Silicon-Gate technology — a technology providing excellent protection against contamination permitting the use of low cost plastic packaging.

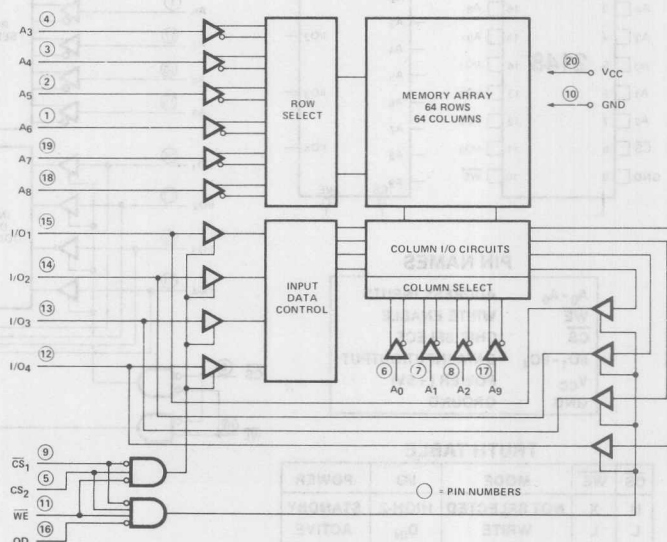
### PIN CONFIGURATION LOGIC SYMBOL



### PIN NAMES

A0-A9	ADDRESS INPUTS	OD	OUTPUT DISABLE
WE	WRITE ENABLE	VCC	POWER (+5V)
CS1, CS2	CHIP SELECT	GND	GROUND
I/O1-I/O4	DATA INPUT/OUTPUT		

### BLOCK DIAGRAM



# 2148

## 1024 × 4 BIT STATIC RAM

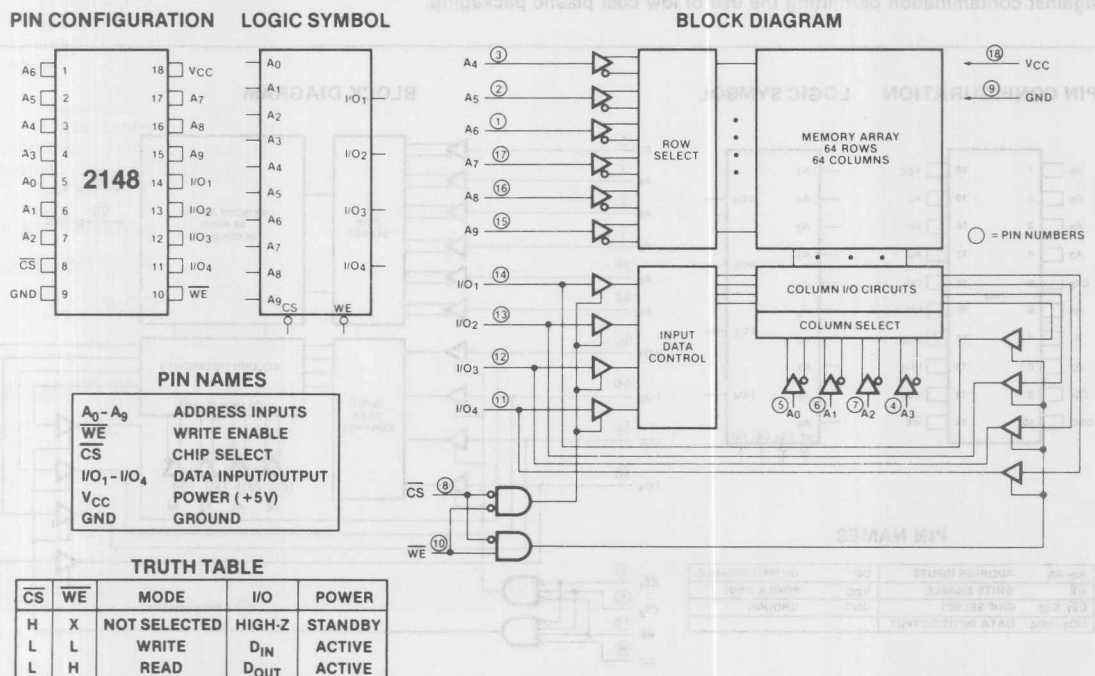
	2148-3	2148	2148-6
Max. Access Time (ns)	55	70	85
Max. Active Current (mA)	125	125	125
Max. Standby Current (mA)	30	30	30

- **HMOS Technology**
- **Completely Static Memory**  
— No Clock or Timing Strobe Required
- **Equal Access and Cycle Times**
- **Single +5V Supply**
- **Automatic Power-Down**
- **High Density 18-Pin Package**
- **Directly TTL Compatible**  
— All Inputs and Outputs
- **Common Data Input and Output**
- **Three-State Output**

The Intel® 2148 is a 4096-bit static Random Access Memory organized as 1024 words by 4 bits using HMOS, a high-performance MOS technology. It uses a uniquely innovative design approach which provides the ease-of-use features associated with non-clocked static memories and the reduced standby power dissipation associated with clocked static memories. To the user this means low standby power dissipation without the need for clocks, address setup and hold times, nor reduced data rates due to cycle times that are longer than access times.

$\overline{CS}$  controls the power-down feature. In less than a cycle time after  $\overline{CS}$  goes high — disabling the 2148 — the part automatically reduces its power requirements and remains in this low power standby mode as long as  $\overline{CS}$  remains high. This device feature results in system power savings as great as 85% in larger systems, where the majority of devices are disabled.

The 2148 is assembled in an 18-pin package configured with the industry standard 1K × 4 pinout. It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. The data is read out nondestructively and has the same polarity as the input data.



## Chapter 6

MCS-85™

MCS-80™

Systems  
Support  
Components

Peripherals

Static RAMs

ROMs-EPROMs

MOS

68000

MOS

68010

ROMs-EPROMs

Static RAMs

Peripherals

Components

Support

Systems

MCS-80<sup>™</sup>

MCS-85<sup>™</sup>

Chapter 6

# 2716

## 16K (2K × 8) UV ERASABLE PROM

### ■ Fast Access Time

- 350 ns Max. 2716-1
- 390 ns Max. 2716-2
- 450 ns Max. 2716
- 650 ns Max. 2716-6

### ■ Single +5V Power Supply

### ■ Low Power Dissipation

- 525 mW Max. Active Power
- 132 mW Max. Standby Power

### ■ Pin Compatible to Intel® 2732 EPROM

### ■ Simple Programming Requirements

- Single Location Programming
- Programs with One 50 ms Pulse

### ■ Inputs and Outputs TTL Compatible during Read and Program

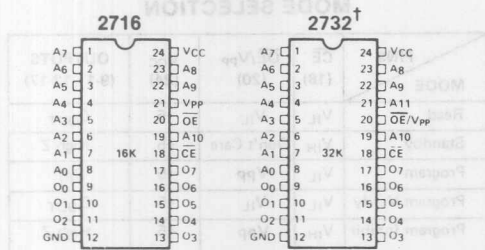
### ■ Completely Static

The Intel® 2716 is a 16,384-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The 2716 operates from a single 5-volt power supply, has a static standby mode, and features fast single address location programming. It makes designing with EPROMs faster, easier and more economical.

The 2716, with its single 5-volt supply and with an access time up to 350 ns, is ideal for use with the newer high performance +5V microprocessors such as Intel's 8085 and 8086. The 2716 is also the first EPROM with a static standby mode which reduces the power dissipation without increasing access time. The maximum active power dissipation is 525 mW while the maximum standby power dissipation is only 132 mW, a 75% savings.

The 2716 has the simplest and fastest method yet devised for programming EPROMs — single pulse TTL level programming. No need for high voltage pulsing because all programming controls are handled by TTL signals. Program any location at any time—either individually, sequentially or at random, with the 2716's single address location programming. Total programming time for all 16,384 bits is only 100 seconds.

### PIN CONFIGURATION



† Refer to 2732 data sheet for specifications

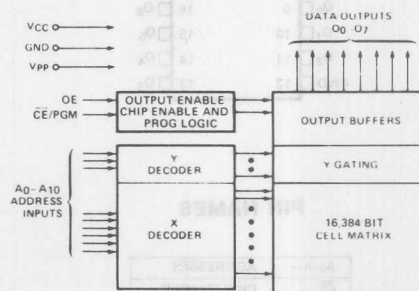
### PIN NAMES

A <sub>0</sub> - A <sub>10</sub>	ADDRESSES
CE/PGM	CHIP ENABLE/PROGRAM
OE	OUTPUT ENABLE
O <sub>0</sub> - O <sub>7</sub>	OUTPUTS

### MODE SELECTION

PINS	CE/PGM (18)	OE (20)	V <sub>pp</sub> (21)	V <sub>CC</sub> (24)	OUTPUTS (9-11, 13-17)
Read	V <sub>IL</sub>	V <sub>IL</sub>	+5	+5	D <sub>OUT</sub>
Standby	V <sub>IH</sub>	Don't Care	+5	+5	High Z
Program	Pulsed V <sub>IL</sub> to V <sub>IH</sub>	V <sub>IH</sub>	+25	+5	D <sub>IN</sub>
Program Verify	V <sub>IL</sub>	V <sub>IL</sub>	+25	+5	D <sub>OUT</sub>
Program Inhibit	V <sub>IL</sub>	V <sub>IH</sub>	+25	+5	High Z

### BLOCK DIAGRAM







## PROGRAMMING

The programming specifications are described in the Data Catalog PROM/ROM Programming Instructions Section.

### ABSOLUTE MAXIMUM RATINGS\*

Temperature Under Bias	-10°C to +80°C
Storage Temperature	-65°C to +125°C
All Input or Output Voltages with Respect to Ground	+6V to -0.3V

#### \*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

## D.C. AND OPERATING CHARACTERISTICS

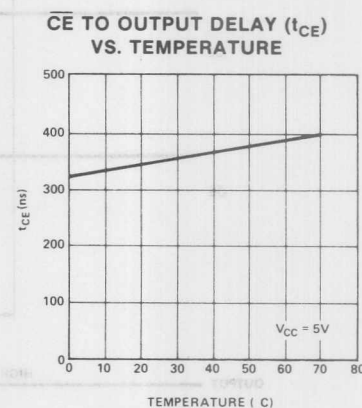
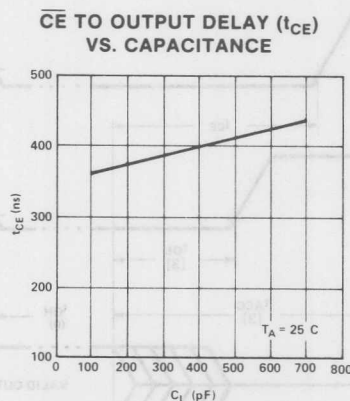
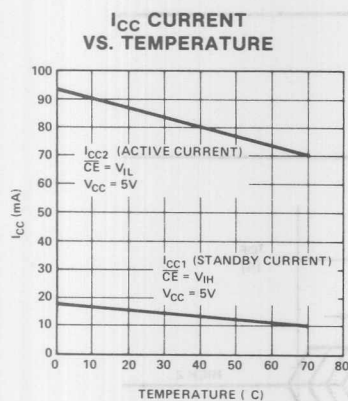
$T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = +5V \pm 5\%$

### READ OPERATION

Symbol	Parameter	Limits			Unit	Conditions
		Min.	Typ. <sup>1</sup>	Max.		
$I_{LI1}$	Input Load Current (except $\overline{OE}/V_{PP}$ )			10	$\mu\text{A}$	$V_{IN} = 5.25V$
$I_{LI2}$	$\overline{OE}/V_{PP}$ Input Load Current			10	$\mu\text{A}$	$V_{IN} = 5.25V$
$I_{LO}$	Output Leakage Current			10	$\mu\text{A}$	$V_{OUT} = 5.25V$
$I_{CC1}$	$V_{CC}$ Current (Standby)		15	30	mA	$\overline{CE} = V_{IH}$ , $\overline{OE} = V_{IL}$
$I_{CC2}$	$V_{CC}$ Current (Active)		85	150	mA	$\overline{OE} = \overline{CE} = V_{IL}$
$V_{IL}$	Input Low Voltage	-0.1		0.8	V	
$V_{IH}$	Input High Voltage	2.0		$V_{CC}+1$	V	
$V_{OL}$	Output Low Voltage			0.45	V	$I_{OL} = 2.1\text{mA}$
$V_{OH}$	Output High Voltage	2.4			V	$I_{OH} = -400\mu\text{A}$

Note: 1. Typical values are for  $T_A = 25^\circ\text{C}$  and nominal supply voltages.

## TYPICAL CHARACTERISTICS



## A.C. CHARACTERISTICS

 $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = +5\text{V} \pm 5\%$ 

Symbol	Parameter	2732 Limits		2732-6 Limits		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
$t_{ACC}$	Address to Output Delay		450		550	ns	$\overline{CE} = \overline{OE} = V_{IL}$
$t_{CE}$	$\overline{CE}$ to Output Delay		450		550	ns	$\overline{OE} = V_{IL}$
$t_{OE}$	Output Enable to Output Delay		120		120	ns	$\overline{CE} = V_{IL}$
$t_{DF}$	Output Enable High to Output Float	0	100	0	100	ns	$\overline{CE} = V_{IL}$
$t_{OH}$	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$ , Whichever Occurred First	0		0		ns	$\overline{CE} = \overline{OE} = V_{IL}$

CAPACITANCE [1]  $T_A = 25^\circ\text{C}$ ,  $f = 1\text{MHz}$ 

Symbol	Parameter	Typ.	Max.	Unit	Conditions
$C_{IN1}$	Input Capacitance Except $\overline{OE}/V_{PP}$	4	6	pF	$V_{IN} = 0\text{V}$
$C_{IN2}$	$\overline{OE}/V_{PP}$ Input Capacitance		20	pF	$V_{IN} = 0\text{V}$
$C_{OUT}$	Output Capacitance		12	pF	$V_{OUT} = 0\text{V}$

## A.C. TEST CONDITIONS

Output Load: 1 TTL gate and  $C_L = 100\text{pF}$ Input Rise and Fall Times:  $\leq 20\text{ns}$ 

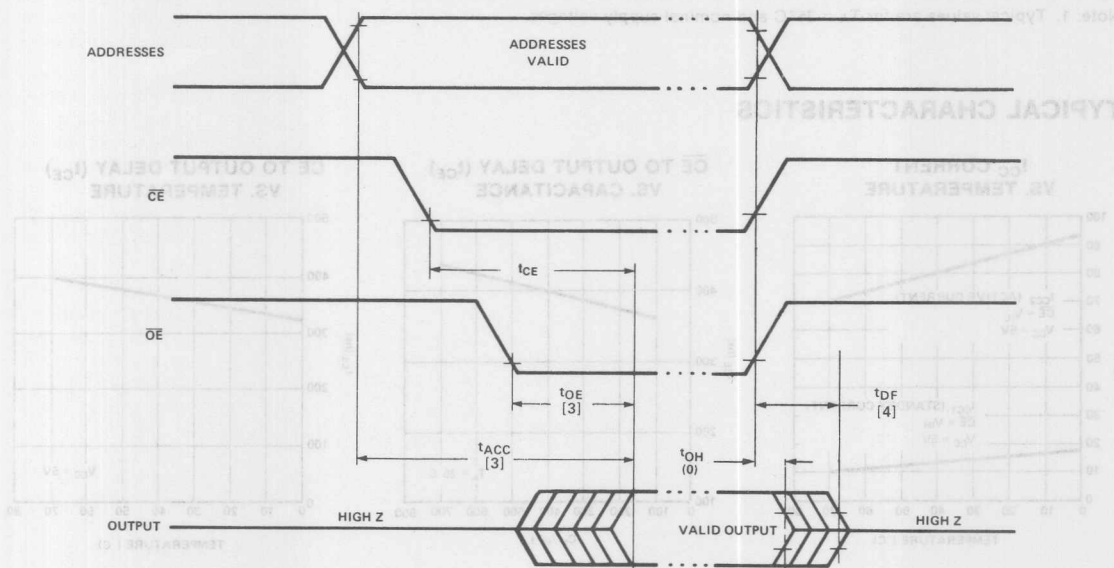
Input Pulse Levels: 0.8V to 2.2V

Timing Measurement Reference Level:

Inputs 1V and 2V

Outputs 0.8V and 2V

## A.C. WAVEFORMS [2]



## NOTES:

1. THIS PARAMETER IS ONLY SAMPLED AND IS NOT 100% TESTED.
2. ALL TIMES SHOWN IN PARENTHESES ARE MINIMUM TIMES AND ARE NSEC UNLESS OTHERWISE SPECIFIED.
3.  $t_{OE}$  MAY BE DELAYED UP TO 330ns AFTER THE FALLING EDGE OF  $\overline{CE}$  WITHOUT IMPACT ON  $t_{ACC}$ .
4.  $t_{DF}$  IS SPECIFIED FROM  $\overline{OE}$  OR  $\overline{CE}$ , WHICHEVER OCCURS FIRST.

## ERASURE CHARACTERISTICS

The erasure characteristics of the 2732 are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms ( $\text{\AA}$ ). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000 $\text{\AA}$  range. Data show that constant exposure to room level fluorescent lighting could erase the typical 2732 in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 2732 is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 2732 window to prevent unintentional erasure.

The recommended erasure procedure (see Data Catalog page 4-83) for the 2732 is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms ( $\text{\AA}$ ). The integrated dose (i.e., UV intensity  $\times$  exposure time) for erasure should be a minimum of 15 W-sec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000  $\mu\text{W}/\text{cm}^2$  power rating. The 2732 should be placed within 1 inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

## DEVICE OPERATION

The five modes of operation of the 2732 are listed in Table 1. A single 5V power supply is required in the read mode. All inputs are TTL levels except for  $\overline{\text{OE}}/V_{\text{PP}}$  during programming. In the program mode the  $\overline{\text{OE}}/V_{\text{PP}}$  input is pulsed from a TTL level to 25V.

TABLE 1. Mode Selection

MODE \ PINS	$\overline{\text{CE}}$ (18)	$\overline{\text{OE}}/V_{\text{PP}}$ (20)	$V_{\text{CC}}$ (24)	OUTPUTS (9-11,13-17)
Read	$V_{\text{IL}}$	$V_{\text{IL}}$	+5	$D_{\text{OUT}}$
Standby	$V_{\text{IH}}$	Don't Care	+5	High Z
Program	$V_{\text{IL}}$	$V_{\text{PP}}$	+5	$D_{\text{IN}}$
Program Verify	$V_{\text{IL}}$	$V_{\text{IL}}$	+5	$D_{\text{OUT}}$
Program Inhibit	$V_{\text{IH}}$	$V_{\text{PP}}$	+5	High Z

### Read Mode

The 2732 has two control functions, both of which must be logically satisfied in order to obtain data at the outputs. Chip Enable ( $\overline{\text{CE}}$ ) is the power control and should be used for device selection. Output Enable ( $\overline{\text{OE}}$ ) is the output control and should be used to gate data to the output pins, independent of device selection. Assuming that addresses are stable, address access time ( $t_{\text{ACC}}$ ) is equal to the delay from  $\overline{\text{CE}}$  to output ( $t_{\text{CE}}$ ). Data is available at the outputs 120ns ( $t_{\text{OE}}$ ) after the falling edge of  $\overline{\text{OE}}$ , assuming that  $\overline{\text{CE}}$  has been low and addresses have been stable for at least  $t_{\text{ACC}} - t_{\text{OE}}$ .

### Standby Mode

The 2732 has a standby mode which reduces the active power current by 80%, from 150mA to 30mA. The 2732 is placed in the standby mode by applying a TTL high signal to the  $\overline{\text{CE}}$  input. When in standby mode, the out-

puts are in a high impedance state, independent of the  $\overline{\text{OE}}$  input.

### Output OR-Tieing

Because EPROMs are usually used in larger memory arrays, Intel has provided a 2 line control function that accommodates this use of multiple memory connections. The two line control function allows for:

- the lowest possible memory power dissipation, and
- complete assurance that output bus contention will not occur.

To most efficiently use these two control lines, it is recommended that  $\overline{\text{CE}}$  (pin 18) be decoded and used as the primary device selecting function, while  $\overline{\text{OE}}$  (pin 20) be made a common connection to all devices in the array and connected to the READ line from the system control bus. This assures that all deselected memory devices are in their low power standby mode and that the output pins are only active when data is desired from a particular memory device.

### Programming

Initially, and after each erasure, all bits of the 2732 are in the "1" state. Data is introduced by selectively programming "0's" into the desired bit locations. Although only "0's" will be programmed, both "1's" and "0's" can be presented in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The 2732 is in the programming mode when the  $\overline{\text{OE}}/V_{\text{PP}}$  input is at 25V. It is required that a 0.1 $\mu\text{F}$  capacitor be placed across  $\overline{\text{OE}}/V_{\text{PP}}$  and ground to suppress spurious voltage transients which may damage the device. The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

When the address and data are stable, a 50msec, active low, TTL program pulse is applied to the  $\overline{\text{CE}}$  input. A program pulse must be applied at each address location to be programmed. You can program any location at any time — either individually, sequentially, or at random. The program pulse has a maximum width of 55msec. The 2732 must not be programmed with a DC signal applied to the  $\overline{\text{CE}}$  input.

Programming of multiple 2732s in parallel with the same data can be easily accomplished due to the simplicity of the programming requirements. Like inputs of the paralleled 2732s may be connected together when they are programmed with the same data. A low level TTL pulse applied to the  $\overline{\text{CE}}$  input programs the paralleled 2732s.

### Program Inhibit

Programming of multiple 2732s in parallel with different data is also easily accomplished. Except for  $\overline{\text{CE}}$ , all like inputs (including  $\overline{\text{OE}}$ ) of the parallel 2732s may be common. A TTL level program pulse applied to a 2732's  $\overline{\text{CE}}$  input with  $\overline{\text{OE}}/V_{\text{PP}}$  at 25V will program that 2732. A high level  $\overline{\text{CE}}$  input inhibits the other 2732s from being programmed.

### Program Verify

A verify should be performed on the programmed bits to determine that they were correctly programmed. The verify is accomplished with  $\overline{\text{OE}}/V_{\text{PP}}$  and  $\overline{\text{CE}}$  at  $V_{\text{IL}}$ . Data should be verified  $t_{\text{PV}}$  after the falling edge of  $\overline{\text{CE}}$ .



2758

## 8K (1K × 8) UV ERASABLE LOW POWER PROM

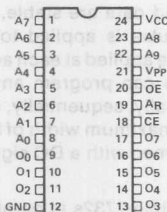
- Single +5V Power Supply
- Simple Programming Requirements
  - Single Location Programming
  - Programs with One 50 ms Pulse
- Low Power Dissipation
  - 525 mW Max. Active Power
  - 132 mW Max. Standby Power
- Fast Access Time: 450 ns Max. in Active and Standby Power Modes
- Inputs and Outputs TTL Compatible during Read and Program
- Completely Static
- Three-State Outputs for OR-Ties

The Intel® 2758 is a 8192-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The 2758 operates from a single 5-volt power supply, has a static standby mode, and features fast single address location programming. It makes designing with EPROMs faster, easier and more economical. The total programming time for all 8192 bits is 50 seconds.

The 2758 has a static standby mode which reduces the power dissipation without increasing access time. The maximum active power dissipation is 525 mW, while the maximum standby power dissipation is only 132 mW, a 75% savings. Power-down is achieved by applying a TTL-high signal to the  $\overline{CE}$  input.

A 2758 system may be designed for total upwards compatibility with Intel's 16K 2716 EPROM (see Applications Note 30). The 2758 maintains the simplest and fastest method yet devised for programming EPROMs — single pulse TTL-level programming. There is no need for high voltage pulsing because all programming controls are handled by TTL signals. Program any location at any time — either individually, sequentially, or at random, with the single address location programming.

### PIN CONFIGURATION



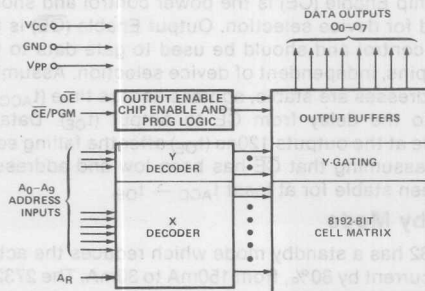
### PIN NAMES

A <sub>0</sub> –A <sub>9</sub>	ADDRESSES
$\overline{CE}/\text{PGM}$	CHIP ENABLE/PROGRAM
OE	OUTPUT ENABLE
O <sub>0</sub> –O <sub>7</sub>	OUTPUTS
AR	SELECT REFERENCE INPUT LEVEL

### MODE SELECTION

PINS	$\overline{CE}/\text{PGM}$ (18)	A <sub>R</sub> (19)	$\overline{OE}$ (20)	V <sub>PP</sub> (21)	V <sub>CC</sub> (24)	OUTPUTS (9-11, 13-17)
Read	V <sub>IL</sub>	V <sub>IL</sub>	V <sub>IL</sub>	+5	+5	D <sub>OUT</sub>
Standby	V <sub>IH</sub>	V <sub>IL</sub>	Don't Care	+5	+5	High Z
Program	Pulsed V <sub>IL</sub> to V <sub>IH</sub>	V <sub>IL</sub>	V <sub>IH</sub>	+25	+5	D <sub>IN</sub>
Program Verify	V <sub>IL</sub>	V <sub>IL</sub>	V <sub>IL</sub>	+25	+5	D <sub>OUT</sub>
Program Inhibit	V <sub>IL</sub>	V <sub>IL</sub>	V <sub>IH</sub>	+25	+5	High Z

### BLOCK DIAGRAM





## 3604A, 3624A FAMILY 4K (512 × 8) HIGH-SPEED PROM

	3604A-2 3624A-2	3604A 3624A	3604AL
Max. $T_A$ (ns)	60	70	90
Max. $I_{CC}$ (mA)	170	170	130/25*

\*Standby Current When The Chip is Deselected.

- Fast Access Time  
--60ns Max (3604A-2, 3624A-2)
- Low Standby Power Dissipation  
(3604AL) --32  $\mu$ W/Bit Max
- Open Collector (3604A)  
or Three State (3624A)  
Outputs
- Four Chip Select Inputs  
For Easy Memory  
Expansion
- Polycrystalline Silicon Fuse  
For Higher Reliability
- Hermetic 24 Pin DIP

The Intel® 3604A/3624A are 4096-bit bipolar PROMs organized as 512 words by 8 bits. The fast second generation 3604A/3624A replaces its Intel predecessor, the 3604/3624. Higher speed PROMs, the 3604A-2/3624A-2, are now available at 60 ns. All 3604A/3624A specifications, except programming, are the same as or better than the 3604/3624. Once programmed, the 3604A/3624A are interchangeable with the 3604/3624.

The PROMs are manufactured with all outputs initially logically high. Logic low levels can be electrically programmed in selected bit locations. Both open collector and three-state outputs are available. Low standby power dissipation can be achieved with the 3604AL. The standby power dissipation is approximately 20% of the active power dissipation.

The 3604A/3624A are available in a hermetic 24-pin dual in-line package. These PROMs are manufactured with the time-proven polycrystalline silicon fuse technology.

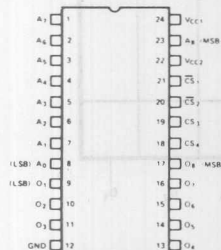
Mode/Pin Connection	Pin 22	Pin 24
READ: 3604A, 3604A-2 3624A, 3624A-2	No Connect or 5V	5V
3604AL	+5V	Must be Left Open
PROGRAM: 3604A, 3604A-2 3624A, 3624A-2	Pulsed 12.5V	Pulsed 12.5V
3604AL	Pulsed 12.5V	Pulsed 12.5V
STANDBY: 3604AL	Power dissipation is automatically reduced whenever the 3604AL is deselected.	

### PIN NAMES

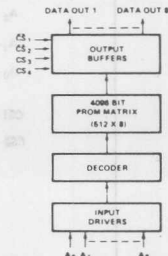
$A_0$ – $A_8$	ADDRESS INPUTS
$CS_1$ – $CS_2$ $CS_3$ – $CS_4$	CHIP SELECT INPUTS <sup>[1]</sup>
$O_1$ – $O_8$	DATA OUTPUTS

[1] To select the PROM  $\overline{CS}_1 = \overline{CS}_2 = 0$   
and  $CS_3 = CS_4 = 1$ .

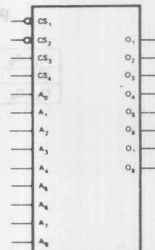
### PIN CONFIGURATION



### BLOCK DIAGRAM



### LOGIC SYMBOL





## 3605A, 3625A 4K (1K x 4) PROM

3605A-1, 3625A-1	50 ns Max.
3605A, 3625A	60 ns Max.

- $\pm 10\%$  Power Supply Tolerance
- Fast Access Time: 40 ns Typically
- Lower Power Dissipation: 0.14 mW/Bit Typically
- Simple Memory Expansion Two Chip Select Inputs
- Open Collector (3605A) and Three-State (3625A) Outputs
- Polycrystalline Silicon Fuse for Higher Reliability
- Hermetic 18-Pin DIP

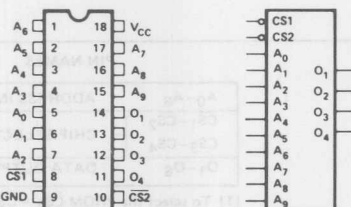
The Intel® 3605A and 3625A families are high density, 4096-bit bipolar PROMs organized as 1024 words by 4 bits. The 1024 by 4 organization gives ideal word or bit modularity for memory array expansion. The 3605A has open collector outputs and the 3625A has three-state outputs. The 3605A and 3625A are fully specified over the 0°C to 75°C temperature range with  $\pm 10\%$  power supply variation. Maximum access times of 50 ns (3605A-2/3625A-2) and 60 ns (3605A/3625A) are available at a typical power dissipation of 0.14 mW/bit.

The 3605A/3625A are packaged in an 18-pin dual in-line hermetic package with 300 milli-inch centers. Thus, twice the bit density can be achieved with the 3605A/3625A in the same memory board areas as 512 by 8-bit PROMs in 24-pin packages.

The highly reliable polycrystalline silicon fuse technology is used in the manufacturing of the 3605A and 3625A families. All outputs are initially a logical high and logic low levels can be electrically programmed in selected bit locations.

### PIN CONFIGURATION

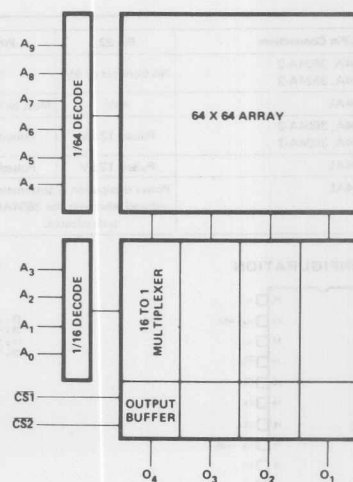
### LOGIC SYMBOL



### PIN NAMES

A <sub>0</sub> - A <sub>8</sub>	ADDRESS INPUTS
CS	CHIP SELECT INPUT
O <sub>1</sub> - O <sub>4</sub>	OUTPUTS

### BLOCK DIAGRAM



# A. C. Characteristics $V_{CC} = +5V \pm 10\%$ , $T_A = 0^\circ C$ to $+75^\circ C$

Symbol	Parameter	Max. Limits		Unit	Conditions
		3605A-1 3625A-1	3605A 3625A		
$t_{A++}, t_{A--}$ $t_{A+-}, t_{A-+}$	Address to Output Delay	50	60	ns	$\overline{CS}_1 = \overline{CS}_2 = V_{IL}$ to select the PROM.
$t_{S++}$	Chip Select to Output Delay	30	30	ns	
$t_{S--}$	Chip Select to Output Delay	30	30	ns	

## Capacitance <sup>(1)</sup> $T_A = 25^\circ C$ , $f = 1$ MHz

SYMBOL	PARAMETER	LIMITS		UNIT	TEST CONDITIONS
		TYP.	MAX.		
$C_{INA}$	Address Input Capacitance	3	8	pF	$V_{CC} = 5V$ $V_{IN} = 2.5V$
$C_{INS}$	Chip-Select Input Capacitance	4	8	pF	$V_{CC} = 5V$ $V_{IN} = 2.5V$
$C_{OUT}$	Output Capacitance	5	10	pF	$V_{CC} = 5V$ $V_{OUT} = 2.5V$

NOTE 1: This parameter is only periodically sampled and is not 100% tested.

## Switching Characteristics

### Conditions of Test:

Input pulse amplitudes: 2.5V

Input pulse rise and fall times of

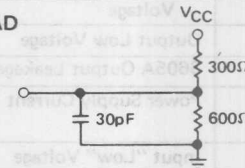
5 nanoseconds between 1 volt and 2 volts

Speed measurements are made at 1.5 volt levels

Output loading is 15 mA and 30 pF

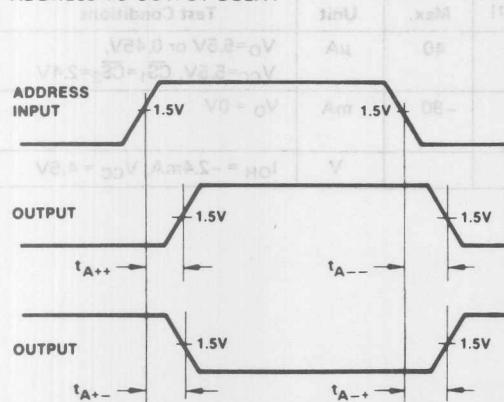
Frequency of test: 2.5 MHz

### 15mA TEST LOAD

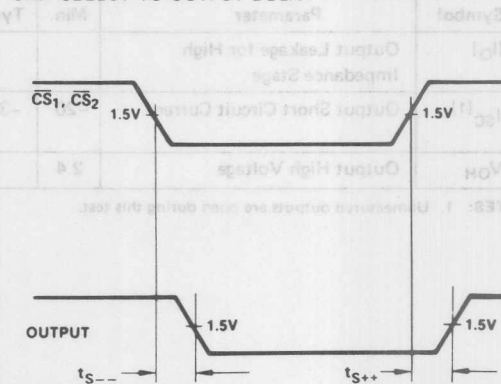


## Waveforms

### ADDRESS TO OUTPUT DELAY



### CHIP SELECT TO OUTPUT DELAY



## PROGRAMMING

The programming specifications are described in the Data Catalog PROM/ROM Programming Instructions on page 4-89.

## Absolute Maximum Ratings\*

Temperature Under Bias	-65°C to +125°C
Storage Temperature	-65°C to +160°C
Output or Supply Voltages	-0.5V to 7 Volts
All Input Voltages	-1V to 5.5V
Output Currents	100mA

## \*COMMENT

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied.

D. C. Characteristics: All Limits Apply for  $V_{CC} = +5.0V \pm 10\%$ ,  $T_A = 0^\circ C$  to  $+75^\circ C$ 

Symbol	Parameter	Limits				Test Conditions
		Min.	Typ. <sup>[1]</sup>	Max.	Unit	
$I_{FA}$	Address Input Load Current		-0.05	-0.25	mA	$V_{CC}=5.5V$ , $V_A=0.45V$
$I_{FS}$	Chip Select Input Load Current		-0.05	-0.25	mA	$V_{CC}=5.5V$ , $V_S=0.45V$
$I_{RA}$	Address Input Leakage Current			40	$\mu A$	$V_{CC}=5.5V$ , $V_A = 5.5V$
$I_{RS}$	Chip Select Input Leakage Current			40	$\mu A$	$V_{CC}=5.5V$ , $V_S = 5.5V$
$V_{CA}$	Address Input Clamp Voltage		-0.9	-1.5	V	$V_{CC}=4.5V$ , $I_A=-10mA$
$V_{CS}$	Chip Select Input Clamp Voltage		-0.9	-1.5	V	$V_{CC}=4.5V$ , $I_S=-10mA$
$V_{OL}$	Output Low Voltage		0.3	0.45	V	$V_{CC}=4.5V$ , $I_{OL}=15mA$
$I_{CEX}$	3605A Output Leakage Current			40	$\mu A$	$V_{CC}=5.5V$ , $V_{CE}=5.5V$
$I_{CC}$	Power Supply Current		110	140	mA	$V_{CC}=5.5V$ , $V_{AO} \rightarrow V_{A9}=0V$ , $\overline{CS}_1=\overline{CS}_2=V_{IH}$
$V_{IL}$	Input "Low" Voltage			0.85	V	
$V_{IH}$	Input "High" Voltage	2.0			V	

## 3625, 3625-2 ONLY

Symbol	Parameter	Min.	Typ. <sup>[1]</sup>	Max.	Unit	Test Conditions
$I_{OL}$	Output Leakage for High Impedance Stage			40	$\mu A$	$V_O=5.5V$ or $0.45V$ , $V_{CC}=5.5V$ , $\overline{CS}_1=\overline{CS}_2=2.4V$
$I_{SC}^{[1]}$	Output Short Circuit Current	-20	-35	-80	mA	$V_O = 0V$
$V_{OH}$	Output High Voltage	2.4			V	$I_{OH} = -2.4mA$ , $V_{CC} = 4.5V$

NOTES: 1. Unmeasured outputs are open during this test.

3628

# 8K (1K X 8) BIPOLAR PROM

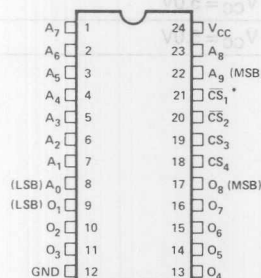
3628	80 ns Max.
3628-4	100 ns Max.

- Fast Access Time: 65 ns Typically
- Low Power Dissipation: 0.09mW/Bit Typically
- Four Chip Select Inputs for Easy Memory Expansion
- Three-State Outputs
- Hermetic 24-Pin DIP
- Polycrystalline Silicon Fuses for Higher Fuse Reliability

The Intel® 3628 is a fully decoded 8192-bit PROM organized as 1024 words by 8 bits. The worst case access time of 80 ns is specified over the 0°C to 75°C temperature range and 5%  $V_{CC}$  power supply tolerances. There are four chip selects provided to facilitate expansion into larger PROM arrays. It uses Schottky clamped TTL technology with polycrystalline silicon fuses. All outputs are initially high and logic low levels can be electrically programmed in selected bit locations.

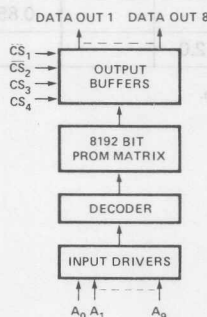
Prior to the 8192 bit 3628, the highest density bipolar PROM available was 4096 bits. The high density of the 3628 now easily doubles the capacity without an increase in area on existing designs currently using 512 words by 8 bit PROMs. There is also little, if any, penalty in power since the 3628 power/bit is approximately one-half that of 4K PROMs. The 3628 is packaged in a hermetic 24-pin dual in-line package.

PIN CONFIGURATION

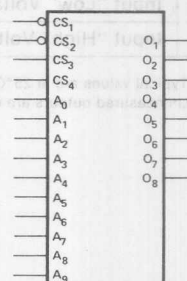


\*PROGRAMMING PIN

BLOCK DIAGRAM



LOGIC SYMBOL



PIN NAMES

$A_0 - A_9$	ADDRESS INPUTS
$CS_1 - CS_4$	CHIP SELECT INPUTS <sup>[1]</sup>
$O_1 - O_8$	DATA OUTPUTS

[1] To select the PROM  $CS_1 = CS_2 = V_{IL}$  and  $CS_3 = CS_4 = V_{IH}$



Temperature Under Bias	-65°C to +125°C
Storage Temperature	-65°C to +160°C
Output or Supply Voltages	-0.5V to 7 Volts
All Input Voltages	-1V to 5.5V
Output Currents	100mA

## \*COMMENT

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied.

**D.C. CHARACTERISTICS:** All Limits Apply for  $V_{CC} = +5.0V \pm 5\%$ ,  $T_A = 0^\circ C$  to  $+75^\circ C$ 

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ. <sup>[1]</sup>	Max.		
$I_{FA}$	Address Input Load Current		-0.05	-0.25	mA	$V_{CC} = 5.25V$ , $V_A = 0.45V$
$I_{FS}$	Chip Select Input Load Current		-0.05	-0.25	mA	$V_{CC} = 5.25V$ , $V_S = 0.45V$
$I_{RA}$	Address Input Leakage Current			40	$\mu A$	$V_{CC} = 5.25V$ , $V_A = 5.25V$
$I_{RS}$	Chip Select Input Leakage Current			40	$\mu A$	$V_{CC} = 5.25V$ , $V_S = 5.0V$
$ I_{OL} $	Output Leakage for High Impedance State			100	$\mu A$	$V_O = 5.25V$ or $0.45V$ , $V_{CC} = 5.25V$ , $CS_1 = CS_2 = 2.4V$
$I_{SC}^{[2]}$	Output Short Circuit Current	-20	-35	-80	mA	$V_O = 0V$
$V_{CA}$	Address Input Clamp Voltage		-0.9	-1.5	V	$V_{CC} = 4.75V$ , $I_A = -10mA$
$V_{CS}$	Chip Select Input Clamp Voltage		-0.9	-1.5	V	$V_{CC} = 4.75V$ , $I_S = -10mA$
$V_{OL}$	Output Low Voltage		0.3	0.45	V	$V_{CC} = 4.75V$ , $I_{OL} = 10mA$
$V_{OH}$	Output High Voltage	2.4	3.4		V	$I_{OH} = -2.4mA$ , $V_{CC} = 4.75V$
$I_{CC}$	Power Supply Current		150	190	mA	$V_{CC} = 5.25V$ , $V_{A0} \rightarrow V_{A9} = 0V$ , PROM deselected
$V_{IL}$	Input "Low" Voltage			0.85	V	$V_{CC} = 5.0V$
$V_{IH}$	Input "High" Voltage	2.0			V	$V_{CC} = 5.0V$

- NOTES: 1. Typical values are at 25°C and at nominal voltage.  
2. Unmeasured outputs are open during this test.

# A.C. CHARACTERISTICS

$V_{CC} = +5V \pm 5\%$ ,  $T_A = 0^\circ C$  to  $+75^\circ C$

SYMBOL	PARAMETER	MAX. LIMITS		UNIT	CONDITIONS
		3628	3628-4		
$t_A$	Address to Output Delay	80	100	ns	$\overline{CS}_1 = \overline{CS}_2 = V_{IL}$ and $CS_3 = CS_4 = V_{IH}$ to select the PROM.
$t_{EN}$	Output Enable Time	40	45	ns	
$t_{DIS}$	Output Disable Time	40	45	ns	

## CAPACITANCE<sup>(1)</sup> $T_A = 25^\circ C$ , $f = 1$ MHz

SYMBOL	PARAMETER	TYP. LIMITS		UNIT	TEST CONDITIONS
		TYP.	MAX.		
$C_{INA}$	Address Input Capacitance	4	10	pF	$V_{CC} = 5V$ $V_{IN} = 2.5V$
$C_{INS}$	Chip-Select Input Capacitance	6	10	pF	$V_{CC} = 5V$ $V_{IN} = 2.5V$
$C_{OUT}$	Output Capacitance	7	15	pF	$V_{CC} = 5V$ $V_{OUT} = 2.5V$

NOTE 1: This parameter is only periodically sampled and is not 100% tested.

## SWITCHING CHARACTERISTICS

### Conditions of Test:

Input pulse amplitudes - 2.5V

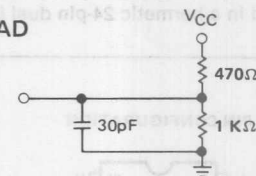
Input pulse rise and fall times of  
5 nanoseconds between 1 volt and 2 volts

Speed measurements are made at 1.5 volt levels

Output loading is 10 mA and 30 pF

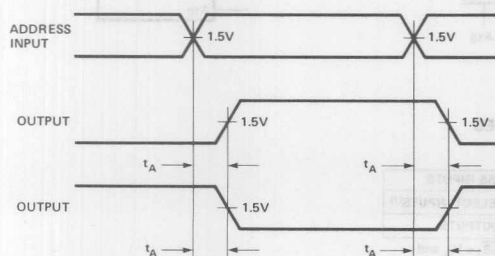
Frequency of test - 2.5 MHz

10 mA TEST LOAD

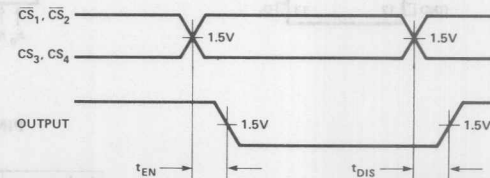


## WAVEFORMS

### ADDRESS TO OUTPUT DELAY



### CHIP SELECT TO OUTPUT DELAY





# 3636 16K (2K × 8) BIPOLAR PROM

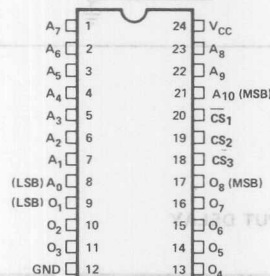
SYMBOL	PARAMETER	UNIT	3636-1	3636
$t_A$	Address to Output Delay	ns	65 ns Max.	80 ns Max.
$t_{EH}$	Output Enable Time	ns		
$t_{DS}$	Output Disable Time	ns		

- Fast Access Time: 50 ns Typically
- Three-State Outputs
- Low Power Dissipation: 0.05 mW/Bit Typically
- Hermetic 24-Pin DIP
- Three Chips Select Input for Easy Memory Expansion
- Polycrystalline Silicon Fuses for Higher Fuse Reliability

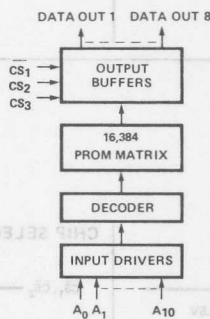
The Intel® 3636 is a fully decoded 16,384 bit PROM organized as 2048 words by 8 bits. The worst case access time of 65 ns is specified over the 0°C to 75°C temperature range and 10%  $V_{CC}$  power supply tolerances. There are three chip selects provided to facilitate expansion into larger PROM arrays. The PROMs use the Schottky clamped TTL technology with polycrystalline silicon fuses. All outputs are initially high and logic low levels can be electrically programmed in selected bit locations.

Prior to the 16,384 bit 3636, the highest density bipolar PROM available was 8192 bits. The high density of the 3636 now easily doubles the capacity without an increase in area on existing designs currently using 1024 by 8 bit PROMs. There is also little, if any, penalty in power since the power/bit is approximately one-half that of 8K PROMs. The 3636 is packaged in a hermetic 24-pin dual in-line package.

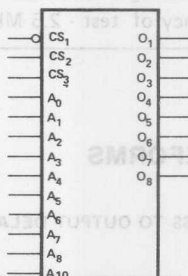
PIN CONFIGURATION



BLOCK DIAGRAM



LOGIC SYMBOL



PIN NAMES

A <sub>0</sub> -A <sub>10</sub>	ADDRESS INPUTS
CS <sub>1</sub> , CS <sub>2</sub> , CS <sub>3</sub>	CHIP SELECT INPUTS <sup>(1)</sup>
O <sub>1</sub> -O <sub>8</sub>	DATA OUTPUTS

(1) To select the PROM CS<sub>1</sub> = V<sub>IL</sub> and CS<sub>2</sub> = CS<sub>3</sub> = V<sub>IH</sub>

## PROGRAMMING

The programming specifications are described in the PROM Programming Section of the Data Catalogue.

## ABSOLUTE MAXIMUM RATINGS\*

Temperature Under Bias	-65°C to +125°C
Storage Temperature	-65°C to +160°C
Output or Supply Voltages	-0.5V to 7 Volts
All Input Voltages	-1V to 5.5V
Output Currents	100mA

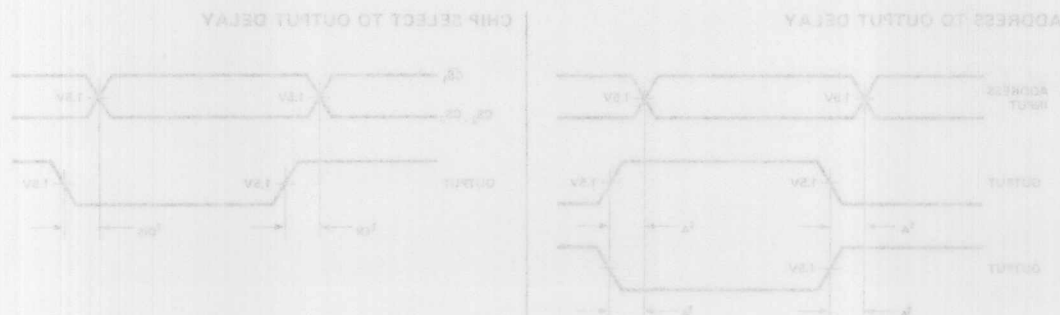
## \*COMMENT

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied.

D.C. CHARACTERISTICS: All Limits Apply for  $V_{CC} = +5.0V \pm 10\%$ ,  $T_A = 0^\circ\text{C}$  to  $+75^\circ\text{C}$ 

Symbol	Parameter	Limits				Test Conditions
		Min.	Typ. <sup>[1]</sup>	Max.	Unit	
$I_{FA}$	Address Input Load Current		-0.05	-0.25	mA	$V_{CC} = 5.5V$ , $V_A = 0.45V$
$I_{FS}$	Chip Select Input Load Current		-0.05	-0.25	mA	$V_{CC} = 5.5V$ , $V_S = 0.45V$
$I_{RA}$	Address Input Leakage Current			40	$\mu\text{A}$	$V_{CC} = 5.5V$ , $V_A = 5.5V$
$I_{RS}$	Chip Select Input Leakage Current			40	$\mu\text{A}$	$V_{CC} = 5.5V$ , $V_S = 5.5V$
$ I_{OL} $	Output Leakage for High Impedance State			100	$\mu\text{A}$	$V_O = 5.5V$ or $0.45V$ , $V_{CC} = 5.5V$ , $CS_1 = 2.4V$
$I_{SC}^{[2]}$	Output Short Circuit Current	-20	-40	-80	mA	$V_O = 0V$
$V_{CA}$	Address Input Clamp Voltage		-0.9	-1.5	V	$V_{CC} = 4.5V$ , $I_A = -10\text{ mA}$
$V_{CS}$	Chip Select Input Clamp Voltage		-0.9	-1.5	V	$V_{CC} = 4.5V$ , $I_S = -10\text{ mA}$
$V_{OH}$	Output High Voltage	2.4	3.2		V	$I_{OH} = -2.4\text{ mA}$ , $V_{CC} = 4.5V$
$V_{OL}$	Output Low Voltage		0.3	0.45	V	$V_{CC} = 4.5V$ , $I_{OL} = 10\text{ mA}$
$I_{CC}$	Power Supply Current		150	185	mA	$V_{CC} = 5.5V$
$V_{IL}$	Input "Low" Voltage			0.85	V	$V_{CC} = 5.0V \pm 10\%$
$V_{IH}$	Input "High" Voltage	2.0			V	$V_{CC} = 5.0V \pm 10\%$

NOTES: 1. Typical values are for  $T_A = 25^\circ\text{C}$  and nominal supply voltages.  
2. Unmeasured outputs are open during this test.



**A.C. CHARACTERISTICS**  $V_{CC} = \pm 5V \pm 10\%$ ,  $T_A = 0^\circ\text{C}$  to  $+75^\circ\text{C}$ 

SYMBOL	PARAMETER	MAX. LIMITS		UNIT	CONDITIONS
		3636-1	3636		
$t_A$	Address to Output Delay	65	80	ns	$\overline{CS}_1 = V_{IL}$ and $CS_2 = CS_3 = V_{IH}$ to select the PROM.
$t_{EN}$	Output Enable Time	40	50	ns	
$t_{DIS}$	Output Disable Time	40	50	ns	

**CAPACITANCE** <sup>(1)</sup>  $T_A = 25^\circ\text{C}$ ,  $f = 1\text{ MHz}$ 

SYMBOL	PARAMETER	TYP. LIMITS		UNIT	TEST CONDITIONS
		TYP.	MAX.		
$C_{INA}$	Address Input Capacitance	4	10	pF	$V_{CC} = 5V$ , $V_{IN} = 2.5V$
$C_{INS}$	Chip-Select Input Capacitance	6	10	pF	$V_{CC} = 5V$ , $V_{IN} = 2.5V$
$C_{OUT}$	Output Capacitance	7	12	pF	$V_{CC} = 5V$ , $V_{OUT} = 2.5V$

NOTE 1: This parameter is only periodically sampled and is not 100% tested.

**SWITCHING CHARACTERISTICS****Conditions of Test:**

Input pulse amplitudes: 2.5V

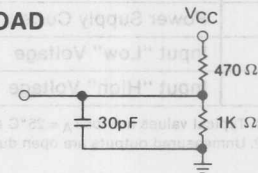
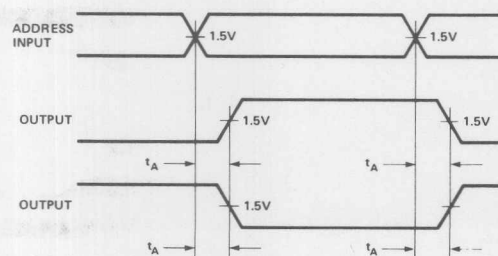
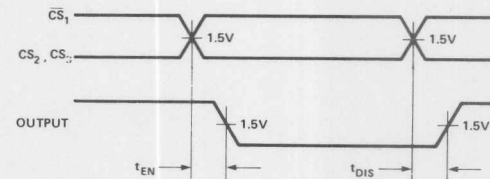
Input pulse rise and fall times of

5 nanoseconds between 1 volt and 2 volts

Speed measurements are made at 1.5 volt levels

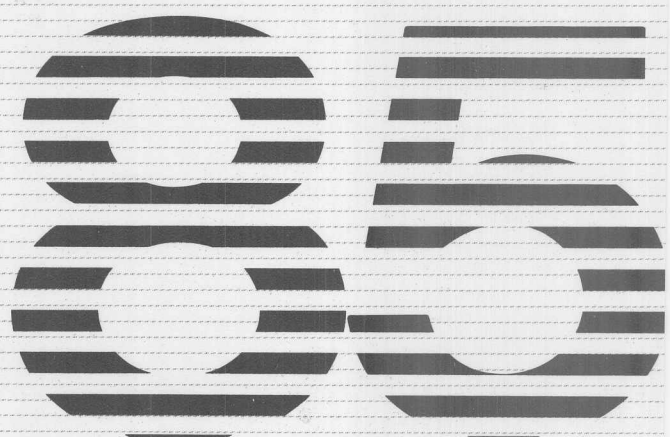
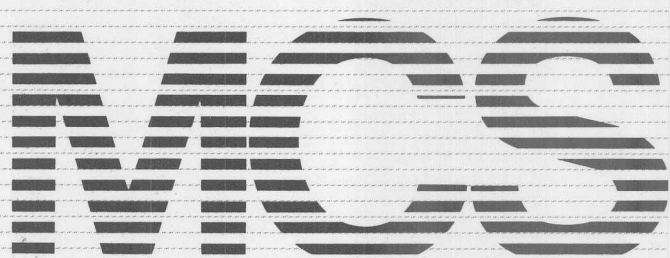
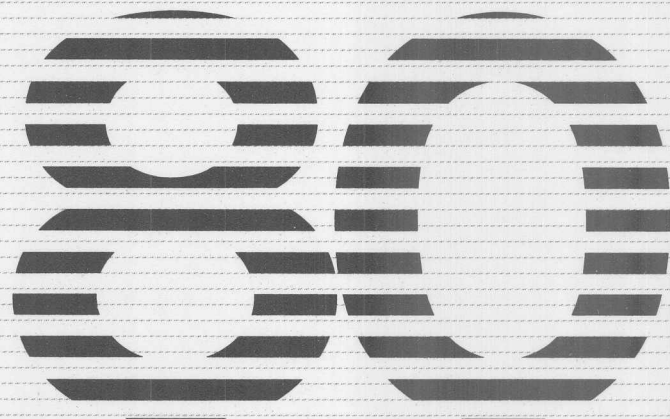
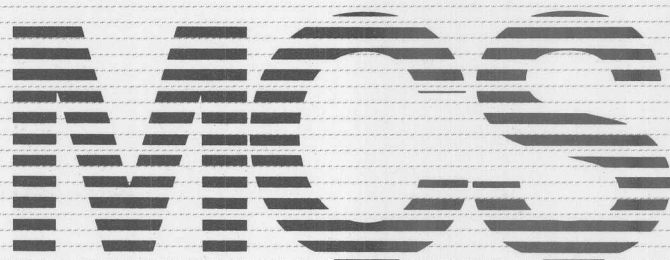
Output loading is 10 mA and 30 pF

Frequency of test: 2.5 MHz

**10 mA TEST LOAD****WAVEFORMS****ADDRESS TO OUTPUT DELAY****CHIP SELECT TO OUTPUT DELAY**



# Chapter 7 Development Aids



# Aids Development Chapter 7



## ICE-80™ 8080 IN-CIRCUIT EMULATOR

**Connects Intellec® system to user configured system via an external cable and 40-pin plug, replacing the user system 8080**

**Allows real-time (2 MHz) emulation of user system 8080**

**Shares Intellec® RAM, ROM, and PROM memory and Intellec® I/O facilities with user system**

**Checks for up to three hardware and four software break conditions**

**Offers full symbolic debugging capabilities**

**Eliminates need for extraneous debugging tools residing in user system**

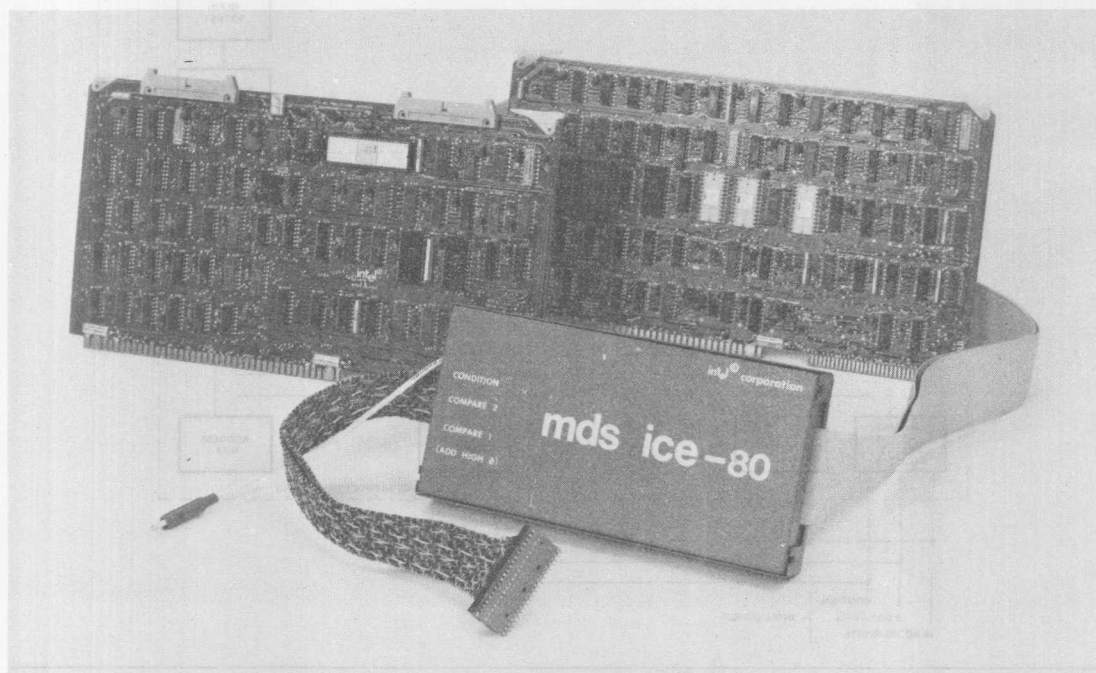
**Provides address, data, and 8080 status information on last 44 machine cycles emulated**

**Provides capability to examine and alter CPU registers, main memory, pin, and flag values**

**Integrates hardware and software development efforts**

**Available in diskette or paper tape versions**

The Intellec ICE-80 8080 In-Circuit Emulator is an Intellec resident module designed to interface with any user configured 8080 system. With ICE-80 as a replacement for a prototype system 8080, the designer may emulate the system's 8080 in real time, single step the system's program, and substitute Intellec memory and I/O for user system equivalents. Powerful Intellec debug functions are extended into the user system. For the first time the designer may examine and modify his system with symbolic references instead of absolute values.



## Integrated Hardware/Software Development

Use of the ICE-80 module enables the system integration phase, which can be so costly and frustrating when attempting to mesh completed hardware and software products, to become a convenient two-way debug tool when begun early in the design cycle. The user prototype need consist of no more than an 8080 CPU socket and a user bus to begin integration of software and hardware development efforts. With the ICE-80 mapping capabilities, system resources may be accessed for missing prototype hardware. Hardware designs may be tested using system software to drive the final product. A functional block diagram of the ICE-80 module is shown in Figure 7-1.

## Symbolic Debugging Capability

ICE-80 provides for user-defined symbolic references to program memory addresses and data. Symbols may be substituted for numeric values in any of the ICE-80 commands. The user is thus relieved from looking up addresses of variables or program subroutines.

**Symbol Table** — The user symbol table generated along with the object file during a PL/M-80 compilation or a MAC80 or resident assembly, is loaded to memory along with the user program to be emulated. The user may add to this symbol table any additional symbolic values for memory addresses, constants, or variables found useful

ing, changing, or breaking at the intended location.

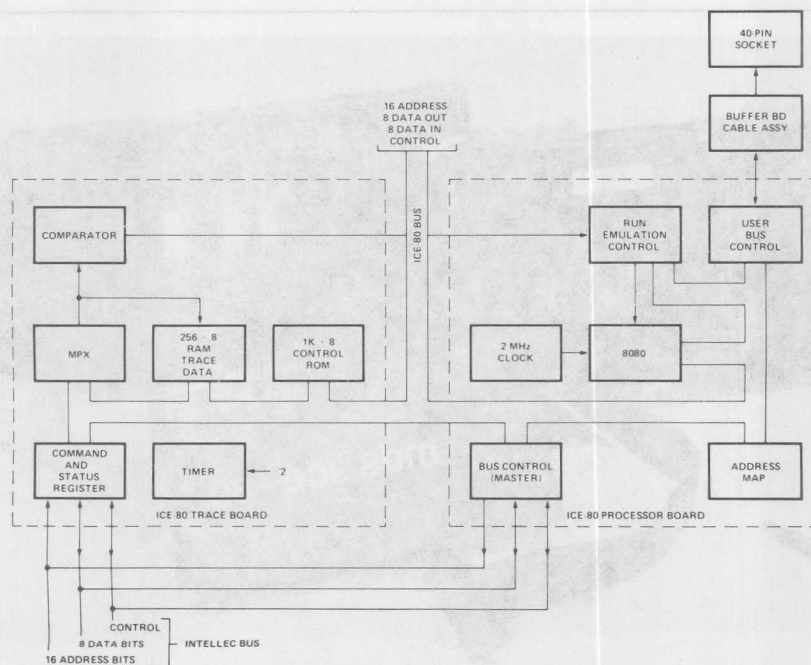
**Symbolic Reference** — ICE-80 provides symbolic definition of all 8080 registers, flags, and selected pins. The following symbolic references are also provided for user convenience: TIMER, a 16-bit register containing the number of  $\phi_2$  clock pulses elapsed during emulation; ADDRESS, the address of the last instruction emulated; INTERRUPTENABLED, the user 8080 interrupt mechanism status; and UPPERLIMIT, the highest RAM address occupied by user memory.

## Debug Capability Inside User System

ICE-80 provides for user debugging of full prototype or production systems without introducing extraneous hardware or software test tools. ICE-80 connects to the user system through the socket provided for the user 8080 in the user system (see Figure 7-2). Intellec memory is used for the execution of the ICE-80 software, while I/O provides the user with the ability to communicate with ICE-80 and receive information on the generation of the user system. A sample ICE-80 debug session is shown in Figure 7-3.

## I/O Mapping and Memory

Memory and I/O for the user system may be resident in the user system or "borrowed" from the Intellec system through ICE-80's mapping capability.



7-1. Functional Block Diagram of ICE-80 Module

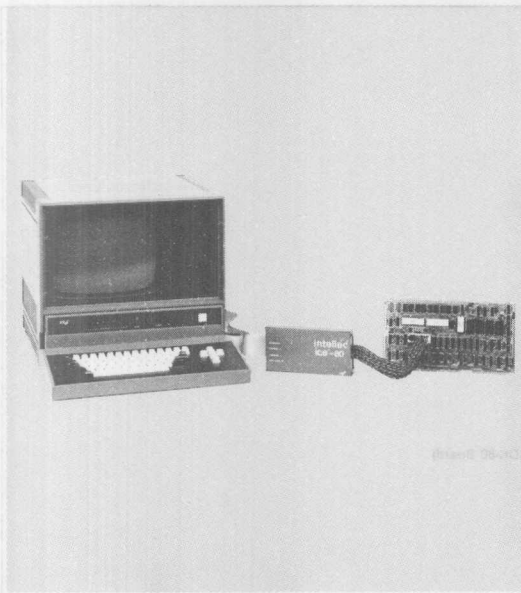


Figure 7-2. ICE-80 Module Installed in User System

**Memory Blocking** — ICE-80 separates user memory into 16 4K blocks. User I/O is divided into 16 16-port blocks. Each block of memory or I/O may be defined independently. The user may assign system equivalents to take the place of devices not yet designed for the user system during prototyping. In addition, memory or I/O may be accessed in place of user system devices during prototype or production checkout.

**Error Messages** — The user may also designate a block of memory or I/O as nonexistent. ICE-80 issues error messages when memory or I/O designated as nonexistent is accessed by the user program.

### Real-Time Trace

ICE-80 captures valuable trace information while the user is executing programs in real time. The 8080 status, the user memory or port addressed, and the data read or written (snap data), is stored for the last 44 machine cycles executed. This provides ample data for determining how the user system was reacting prior to emulation break. It is available whether the break was user initiated or the result of an error condition. For detailed information on the actions of CPU registers, flags, or other system operations, the user may operate in single or multiple step sequences tailored to system debug needs.

### Hardware

The heart of the ICE-80 is a microcomputer system utilizing Intel's 8080 microprocessor as its nucleus. This system communicates with the Intellec host processor via I/O commands. Host processor commands and ICE-80 status are interchanged through registers on the

ICE-80 trace board. ICE-80 and the system also communicate through a control block resident in the Intellec main memory, which contains detailed configuration and status information transmitted at an emulation break. ICE-80 hardware consists of two PC boards — the processor and trace boards residing in the Intellec chassis — and a 6-foot cable interfacing to the user system. The trace and processor boards communicate with the system on the bus, and also with each other on a separate ICE-80 bus. ICE-80 connects to the user system through a cable that plugs directly into the socket provided for the user's 8080.

### Trace Board

The trace board talks to the system as a peripheral device. It receives commands to ICE-80 and returns ICE-80 responses. While ICE-80 is executing the user program, the trace board collects data for each machine cycle emulated (snap data). The information is continuously stored in high-speed bipolar memory.

**Breakpoint** — The trace board also contains two 24-bit hardware breakpoint registers which can be loaded by the user. While in emulation mode, a hardware comparator is constantly monitoring address and status lines for a match to terminate an emulation. A user probe is also available for attachment to any user signal. When this signal goes true a break condition is recognized.

**Interrogation** — The trace board signals the processor board when a command to ICE-80 or break condition has been detected. The ICE-80 CPU then sends data stored on the trace board to the control block in memory. Snap data, along with information on 8080 registers and pin status and the reason for the emulation break, are then available for access during interrogation mode. Error conditions, if present, are transmitted and automatically displayed for the user.

### Processor Board

An 8080 CPU resides on the processor board. During emulation it executes instructions from the user's program. At all other times it executes instructions from the control program in the trace module's ROM.

**Timing** — The processor board contains an internal clock generator to provide clocks to the user emulation CPU at 2 MHz. The CPU can alternately be driven by a clock derived from user system signal lines. The clock source is selected by a jumper option on the board. A timer on the trace board counts the  $\phi_2$  clock pulses during emulation and can provide the user with the exact timing of the emulation.

**On/Off Control** — The processor board turns on an emulation when ICE-80 has received a run command from the system. It terminates emulation when a break condition is detected on the trace board, or the user's program attempts to access memory or I/O ports designated as nonexistent in the user system, or the user 8080 is inactive for a quarter of a second.

**Status Storage** — The address map located on the processor board stores the assigned location of each user memory or I/O block. During emulation the processor board determines whether to send/receive information



ISIS 8080 MACROASSEMBLER, V1.0

PAGE 1

```

;USER PROGRAM TO OUTPUT A SERIES OF
;CHARACTERS TO SDK-80 CONSOLE DEVICE
;
1320      ORG 1320H
01E3     EQU 1E3H ;SDK-80 CONSOLE OUT DRIVER
;
1320 0601 START: MVI B,1 ;SET UP B VALUE
1322 3A3613 LDA DAT1 ;LOAD A WITH DAT1 VALUE
135 4F     LOOP: MDV C,A
1326 CDE301 CALL C0 ;SEND C VALUE TO CONSOLE
1329 79     MOV A,C ;RESTORE A
132A 93     SBB B ;SUBTRACT B FROM A
132B 323713 STA RSLT ;STORE RESULT IN RSLT
132E FE40   CPI 40H ;LAST VALUE TO PRINT
1330 C22513 JNZ LOOP ;LOOP AGAIN IF A>40H
1333 C32013 JMP START ;ELSE RESTART WHOLE PROCEDURE
;
1336 5A     DAT1: DB 5AH
1337       RSLT: DS 1
0000       END

```

ISIS, V1.0 INITIAL ICE-80 SESSION

-ICE80 (Note: The SDK-80 Monitor has already been used to initialize the SDK-80 Board)

ISIS ICE-80, V1.0

① \*\*XFORM MEMORY 0 TO 1 U

\*XFORM IO 0FH U

② \*LOAD PROG. HEX

ERR = 067

STAT = 11H TYPE = 06H CMND = 07H ADDR = 1320H GOOD = 06H BAD = 04H

\*CHANGE MEMORY 1321H = FFH

ERR = 067

STAT = 11H TYPE = 06H CMND = 07H ADDR = 1321H GOOD = FFH BAD = FDH

\*LOAD PROG. HEX

③ \*GO FROM START UNTIL RSLT WRITTEN

EMULATION BEGUN

④ ERR = 067

STAT = 11H TYPE = 07H CMND = 02H

⑤ \*DISPLAY CYCLES 5

STAT = A2H ADDR = 1326H DATA = CDH

STAT = 82H ADDR = 1327H DATA = E3H

STAT = 82H ADDR = 1328H DATA = 01H

STAT = 04H ADDR = FFFFH DATA = 13H

STAT = 04H ADDR = FFFFH DATA = 29H

⑥ \*CHANGE DOUBLE REGISTER SP = 13FFH

\*BASE HEX

\*EQUATE STOP = 1333H

⑦ \*GO FROM START UNTIL STOP EXECUTED THEN DUMP

EMULATION BEGUN

B = 01H C = 41H D = 00H E = 00H H = 00H L = 00H F = 56H A = 40H P = 1320H \* = 1333H S = 13FFH

EMULATION TERMINATED AT 1333H

⑧ \*EXIT

\*FFFF

## Notes

1. Set up user memory and I/O. The program is set up to execute in block 1 (1000H-1FFFFH) of user memory, and requires access to the SDK-80 monitor (block 0) and I/O ports in block 0FH. Both ports and memory are defined as available to the user system. All other memory and I/O is initialized by ICE-80 as nonexistent (guarded).
2. A load command generates an error. The type and command numbers indicate that a data mismatch occurred on a write to memory command. The data to be written to address 1320H should have been 06H. When ICE-80 read the data after writing it, a 04H was detected. A change command to a different memory address hints that bit 1 does not go to 1 anywhere in this memory block. Examination indicates that a pin was shorted on the RAM located at 1300H-13FFFH in the prototype system. The problem is fixed and a subsequent load succeeds.
3. A real-time emulation is begun. The program is executed from 'START' (1320H) and continues until 'RSLT' is written [in location 1328H, the contents of the accumulator is stored in (written into) 'RSLT'].
4. An error condition results: TYPE 07, CMND 02 indicate the program accessed is a guarded area.
5. The last 5 machine cycles executed are displayed. The last instruction executed was a call (CDH). The fourth and fifth cycles are a push operation (designated by status 04H) to store the program counter before executing the call. The stack pointer was not initialized in the program and is accessing memory location FFFFH.
6. After making a note to initialize the stack pointer in the next assembly, a temporary fix is effected by setting the stack pointer to the top of user available memory.
7. After setting the base for displays to hex and adding the symbol 'STOP' to the symbol table, emulation is started which will terminate when the instruction at 1333H ('STOP') is executed. When emulation terminates, a dump of the contents of user 8080 registers is requested. One can see that the value of the accumulator is set at 40H, the stack pointer is set at 13FFH, the last address executed (\*) is 1333H, and the program counter has been set to 1320H.
8. Exit returns control to the MDS monitor.

Figure 7-3. Sample ICE-80 Debug Session

on the Intellec or user bus by consulting the address map. The processor board allows the ICE-80 CPU to gain access to the bus as a master to "borrow" Intellec facilities. At an emulation break, the processor board stores the status of specified 8080 input and output signals, disables all interaction with the user bus, and commands the trace board to send stored information to a control block in Intellec memory for access during interrogation mode.

## Cable Card

The cable card is included for cable driving. It transmits address and data bus information to the user system through a 40-pin connector that plugs into the user system in the socket designed for the 8080 when enabled by the processor module's user bus control logic.

## Software

The ICE-80 software driver is a RAM-based program providing easy to use English language commands for defining breakpoints, initiating emulation, and interrogating and altering the user system status recorded during emulation. ICE-80 commands are configured with

a broad range of modifiers to provide the user with maximum flexibility in describing the operation to be performed. Listings of emulation commands, interrogation commands, and utility commands are provided in Tables 7-1, 7-2 and 7-3, respectively.

Command	Operation
Base	Establishes mode of display for output data.
Display	Prints contents of memory, 8080 registers, input ports, 8080 flags, 8080 pins, snap data, symbol table, or other diagnostic data on list device. May also be used for base-to-base conversion, or for addition or subtraction in any base.
Change	Alters contents of memory, register, output port, or 8080 flag.
XFORM	Defines memory and I/O status.
Search	Looks through memory range for specified value.

Table 7-2. ICE-80 Interrogation Commands

Command	Operation
Go	Initiates real-time emulation and allows user to specify breakpoints, data retrieval, and conditions under which emulation should be reinitiated.
Step	Initiates emulation in single or multiple instruction increments. User may specify register dump or tailor diagnostic activity to his needs following each step, and define conditions under which stepping should continue.
Range	Delimits blocks of instructions for which register dump or tailored diagnostics are to occur.
Continue	Resumes real-time emulation.
Call	Emulates user system interrupt.

Table 7-1. ICE-80 Emulation Commands

Command	Operation
Load	Fetches user symbol table and object code from input device.
Save	Sends user symbol table and object code to output device.
Equate	Enters symbol name and value to user symbol table.
Fill	Fills memory range with specified value.
Move	Moves block of memory data to another area of memory.
Timeout	Enables/disables user CPU 1/4 second wait state timeout.
List	Defines list device (diskette-based version only).
Exit	Returns program control to monitor.

Table 7-3. ICE-80 Utility Commands

## SPECIFICATIONS

### Paper Tape-Based Operating Environment

#### Required Hardware

Intellec system  
System console  
Reader device  
Punch device  
ICE-80 module

#### Required Software

System monitor

### Diskette-Based

### Operating Environment

#### Required Hardware

Intellec system  
32K bytes RAM memory  
System console  
Intellec diskette operating system  
ICE-80 module

#### Required Software

System monitor  
ISIS-II

# ICE-80™ IN-CIRCUIT EMULATOR

## System Clock

Crystal controlled 2.185 MHz  $\pm$  0.01%. May be replaced by user clock through jumper selection.

## Physical Characteristics

Width — 12.00 in. (30.48 cm)

Height — 6.75 in. (17.15 cm)

Depth — 0.50 in. (1.27 cm)

Weight — 8.00 lb (3.64 kg)

## Electrical Characteristics

### DC Power Requirements

V<sub>CC</sub> = + 5V,  $\pm$  5%

I<sub>CC</sub> = 9.81A max; 6.90A typ

V<sub>DD</sub> = + 12V,  $\pm$  5%

I<sub>DD</sub> = 79 mA max; 45 mA typ

V<sub>BB</sub> = - 9V,  $\pm$  5%

I<sub>BB</sub> = 1 mA max; 1 $\mu$ A typ

## Environmental Characteristics

Operating Temperature — 0°C to 40°C

Operating Humidity — Up to 95% relative humidity without condensation

## Equipment Supplied

Printed circuit modules (2)

Interface cables and buffer board

ICE-80 software driver, paper tape version

(ICE-80 software driver, diskette-based version is supplied with diskette operating systems)

Operator's Manual

## ORDERING INFORMATION

### Part Number Description

MDS-80-ICE 8080 CPU in-circuit emulator, cable assembly and interactive software included

Command	Operation
Load	Fetches user symbol table and object code from input device.
Save	Sends user symbol table and object code to output device.
Evaluate	Enters symbol name and value to user symbol table.
Fill	Fills memory range with specified value.
Move	Moves block of memory data to another area of memory.
Timeout	Enables/disables user CPU in second well state timeout.
List	Defines list device (diskette-based version only).
Exit	Returns program control to monitor.

Table 7-5: ICE-80 Utility Commands

Command	Operation
Go	Initiates real-time emulation and allows user to specify breakpoints, datastalls, and conditions under which emulation should be initialized.
Step	Initiates emulation in single or multiple instruction increments. User may specify register dump or failor diagnostic activity to his needs following each step, and defines conditions under which stepping should continue.
Range	Delimits blocks of instructions for which register dump or failor diagnostics are to occur.
Continue	Resumes real-time emulation.
Call	Emulates user system interrupt.

Table 7-1: ICE-80 Emulation Commands

## SPECIFICATIONS

Paper Tape-Based  
Operating Environment  
Required Hardware  
Intellic system  
System console  
Reader device  
Punch device  
ICE-80 module  
Required Software  
System monitor

Diskette-Based  
Operating Environment  
Required Hardware  
Intellic system  
32K bytes RAM memory  
System console  
Intellic diskette operating system  
ICE-80 module  
Required Software  
System monitor  
ISIS-II



UPP-103\*

## UNIVERSAL PROM PROGRAMMER

*\*Replaces UPP-101, UPP-102 Universal PROM Programmers*

**Intellec® development system peripheral  
for PROM programming and verification**

**Provides personality cards for program-  
ming all Intel PROM families**

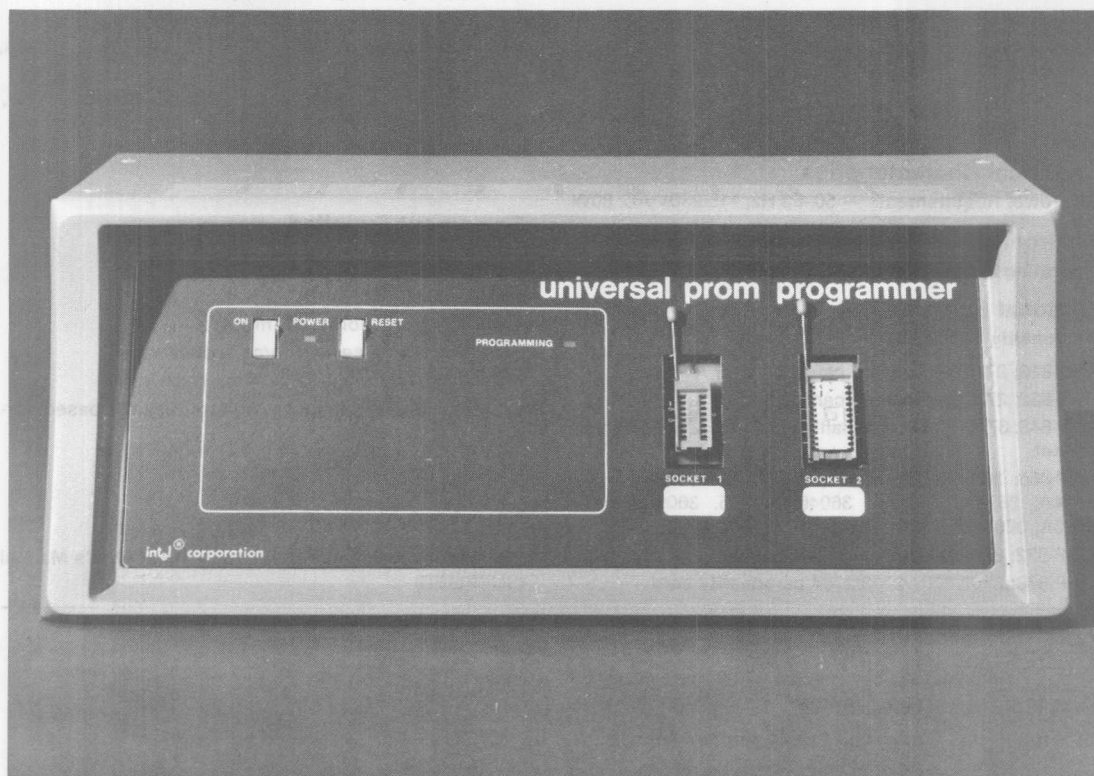
**Provides zero insertion force sockets for  
both 16-pin and 24-pin PROMs**

**Universal PROM mapper software pro-  
vides powerful data manipulation and  
programming commands**

**Provides flexible power source for  
system logic and programming pulse  
generation**

**Holds two personality cards to facilitate  
programming operations using several  
PROM types**

The UPP-103 Universal PROM Programmer is an Intellec system peripheral capable of programming and verifying all of the Intel programmable ROMs (PROMs). In addition, the UPP-103 programs the PROM memory portions of the 8748 microcomputer, 8741 UPI, the 8755 PROM and I/O chip and the 2920 signal processor. Programming and verification operations are initiated from the Intellec development system console and are controlled by the universal PROM mapper (UPM) program.



## FUNCTIONAL DESCRIPTION

### Universal PROM Programmer

The basic Universal PROM Programmer (UPP) consists of a controller module, two personality card sockets, a front panel, power supplies, a chassis, and an Intellec development system interconnection cable. An Intel 4040-based intelligent controller monitors the commands from the Intellec System and controls the data transfer interface between the selected PROM personality card and the Intellec memory. A unique personality card contains the appropriate pulse generation functions for each Intel PROM family. Programming and verifying any Intel PROM may be accomplished by selecting and plugging in the appropriate personality card. The front panel contains a power-on switch and indicator, a reset switch, and two zero-force insertion sockets (one 16-pin and one 24-pin or two 24-pin). A central power supply provides power for system logic and for PROM programming pulse generation. The Universal PROM Programmer may be used as a table top unit or mounted in a standard 19-inch RETMA cabinet.

### Universal PROM Mapper

The Universal PROM Mapper (UPM) is the software program used to control data transfer between paper tape or diskette files and a PROM plugged into the Universal PROM Programmer. It uses Intellec system memory for intermediate storage. The UPM transfers data in 8-bit HEX, BNPF, or binary object format between paper tape or diskette files and the Intellec system memory. While the data is in Intellec system memory, it can be displayed and changed. In addition, word length, bit position, and data sense can be adjusted as required for the PROM to be programmed. PROMs may also be duplicated or altered by copying the PROM contents into the Intellec system memory. Easy to use program and compare commands give the user complete control over programming and verification operations. The UPM eliminates the need for a variety of personalized PROM programming routines because it contains the programming algorithms for all Intel PROM families. The UPM (diskette based version) is included with the Universal PROM Programmer.

## SPECIFICATIONS

### Hardware Interface

**Data** — Two 8-bit unidirectional buses

**Commands** — 3 write commands, 2 read commands, one initiate command

### Physical Characteristics

**Width** — 6 in. (14.7 cm)

**Height** — 7 in. (17.2 cm)

**Depth** — 17 in. (41.7 cm)

**Weight** — 18 lb (8.2 kg)

### Electrical Characteristics

**AC Power Requirements** — 50–60 Hz; 115/230V AC; 80W

### Environmental Characteristics

**Operating Temperature** — 0°C to 55°C

### Optional Equipment

#### Personality Cards

UPP-816: 2716 personality card

UPP-832: 2732 personality card

UPP-848: 8748, 8741 personality card with 40-pin adaptor socket

UPP-865: 3602, 3622, 3602A, 3622A, 3621, 3604, 3624, 3604A, 3624A, 3604AL, 36046-6, 3605, 3605A, 3625, 3625A, 3608, 3628, 3636

UPP-872: 8702A/1702A personality card

UPP-878: 8708/8704/2708/2704 personality card

UPP-955: 8755A personality card with 40-pin adaptor socket

### PROM Programming Sockets

UPP-501: 16-pin/24-pin socket pair

UPP-502: 24-pin/24-pin socket pair

UPP-562: Socket adaptor for 3621, 3602, 3622, 3602A, 3622A

UPP-555: Socket adaptor for 3604AL, 36046-6, 3608, 3628, 3636

UPP-566: Socket adaptor for 3605, 3625, 3605A, 3625A

### Equipment Supplied

Cabinet

Power supplies

4040 intelligent controller module

Specified zero insertion force socket pair

Intellec development system interface cable

Universal PROM Mapper program (diskette-based version)

### Reference Manuals

**9800819** — Universal PROM Programmer User's Manual (SUPPLIED)

## ORDERING INFORMATION

### Part Number Description

UPP-103	Universal PROM programmer with 16-pin/24-pin socket pair and 24-pin/24-pin socket pair.
---------	---





## MODEL 220 INTELLEC® SERIES II MICROCOMPUTER DEVELOPMENT SYSTEM

**Complete microcomputer development system in one package for MCS-86, MCS-85, MCS-80 and MCS-48™ microprocessor families**

**Single LSI electronics board with CPU, 32K bytes RAM memory, and 4K bytes ROM memory**

**Self-test diagnostic capability**

**Eight-level nested, maskable priority interrupt system**

**Built-in interfaces for high speed paper tape reader/punch, printer, and universal PROM programmer**

**Integral CRT with detachable upper/lower case typewriter-style full ASCII keyboard**

**Integral 250K-byte floppy disk with total storage capacity expandable to over 2M bytes**

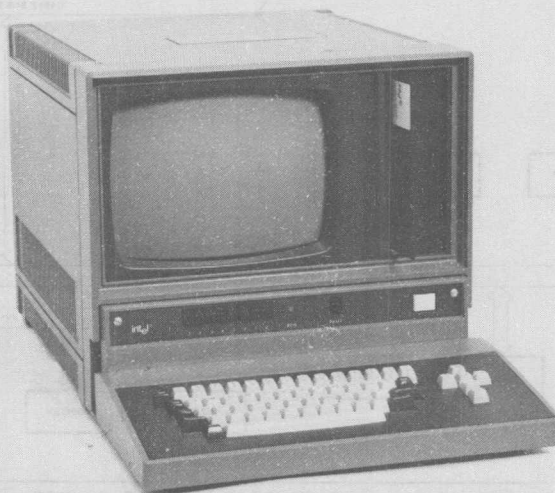
**Powerful ISIS-II Diskette Operating System with relocating macroassembler, linker, and locator**

**Standard MULTIBUS with multi-processor and DMA capability**

**Compatible with standard Intellec®/iSBC™ expansion modules**

**Software compatible with previous Intellec® systems**

The Model 220 Intellec Series II Microcomputer Development System is a complete microcomputer development system integrated into one compact package. It includes a CPU with 32K bytes of RAM memory, 4K bytes of ROM memory, a 2000-character CRT, detachable full ASCII keyboard with cursor controls and upper/lower case capability, and a 250K-byte floppy diskette drive. Powerful ISIS-II Diskette Operating System software allows the Model 220 to be used quickly and efficiently for assembling and debugging programs for Intel's MCS-86, MCS-85, MCS-80, or MCS-48 microprocessor families without the need for paper tape handling. ISIS-II performs all file handling operations for the user, leaving him free to concentrate on the details of his own application. When used in conjunction with an optional in-circuit emulator (ICE) module, the Model 220 provides all the hardware and software development tools necessary for the rapid development of a microcomputer-based product.



## FUNCTIONAL DESCRIPTION

### Hardware Components

The Inteltec Series II Model 220 is a packaged, highly integrated microcomputer development system consisting of a CRT chassis with a 6-slot cardcage, power supply, fans, cables, single floppy diskette drive, and two printed circuit cards. A separate, full ASCII keyboard is connected with a cable.

**CPU Cards** — The master CPU card contains its own microprocessor, memory, I/O, interrupt, and bus interface circuitry, fashioned from Intel's high-technology LSI components. Known as the integrated processor board (IPB), it occupies the first slot in the cardcage. A second, slave CPU card, is responsible for all remaining I/O control, including the CRT and keyboard interface and floppy disk control. This card, mounted on the rear panel, also contains its own microprocessor, RAM and ROM memory, and I/O interface, thus in effect creating a dual processor environment. Known as the I/O controller (IOC), the slave CPU card communicates with the IPB over an 8-bit bidirectional data bus, thus leaving the remaining 5 slots in the cardcage available for system expansion. A block diagram of the IOC is shown in Figure 7-4.

### System Components

The heart of the IPB is an Intel NMOS 8-bit microprocessor, the 8080A-2, running at 2.6 MHz. 32K bytes of RAM memory are provided on the board using Intel 16K RAMs. 4K of ROM is provided, preprogrammed with system bootstrap "self-test" diagnostics and the Inteltec Series II System Monitor. The eight-level vectored priority interrupt system allows interrupts to be individually masked. Using Intel's versatile 8259A interrupt controller, the interrupt system may be user programmed to respond to individual needs.

### Input/Output

**IPB Serial Channels** — The I/O subsystem in the Model 220 consists of two parts: the IOC card and two serial channels on the IPB itself. Each serial channel is RS232 compatible and is capable of running asynchronously from 110 to 9600 baud or synchronously from 150 to 56K baud. Both may be connected to a user defined data set or data terminal. One channel contains current loop adapters. Both channels are implemented using Intel's 8251 USART. They can be programmatically selected to perform a variety of I/O functions. Baud rate selection is accomplished programmatically through an Intel 8253 interval timer. The 8253 also serves as a real-time clock for the entire system. I/O activity through both serial channels is signaled to the system through a second 8259 interrupt controller, operating in a polled mode, nested to the primary 8259.

**IOC Interface** — The remainder of system I/O activity takes place in the IOC. The IOC provides interfaces for the CRT, keyboard, integral floppy disk and standard Inteltec peripherals, including a printer, high speed paper tape reader/punch, and universal PROM programmer. The IOC contains its own independent microprocessor, also an 8080A-2. This CPU controls all I/O operations, as well as supervising communications with the IPB. 8K bytes of ROM contain all I/O control firmware. 8K bytes of ROM are used for CRT screen refresh storage and the floppy disk buffer. These do not occupy any space in Inteltec Series II main memory since the IOC is a totally independent microcomputer subsystem.

**Integral CRT Display** — The CRT is a 12-inch raster scan-type monitor with a 50/60 Hz vertical scan rate and 15.5 kHz horizontal scan rate. Controls are provided for brightness and contrast adjustments. The interface to the CRT is provided through an Intel 8275 single chip, programmable CRT controller. The master processor on

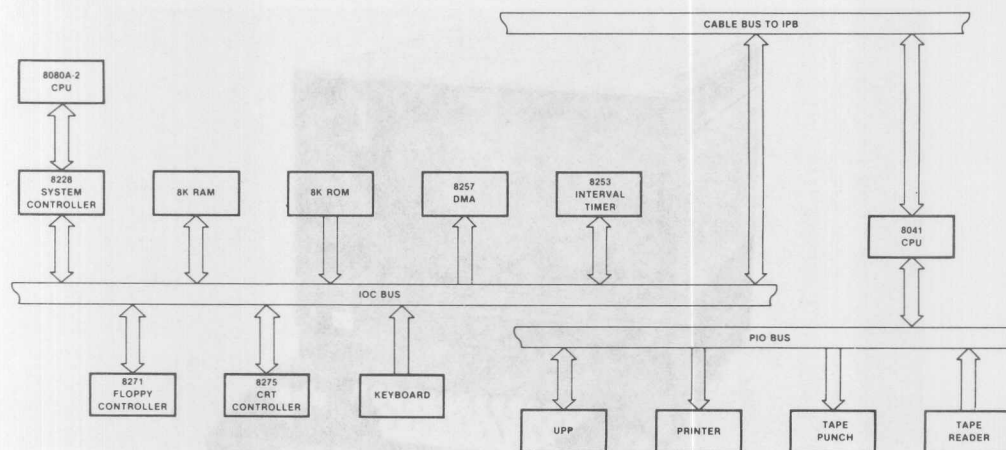


Figure 7-4. I/O Controller (IOC) Block Diagram for the Model 220 Inteltec Series II Microcomputer Development System

the IPB transfers a character for display to the IOC, where it is stored in RAM. The CRT controller reads a line at a time into its line buffer through an Intel 8257 DMA controller and then feeds one character at a time to the character generator to produce the video signal. Timing for the CRT control is provided by an Intel 8253 interval timer. The screen display is formatted as 25 rows of 80 characters. The full set of ASCII characters are displayed, including lower-case alphas.

**Keyboard** — The keyboard interfaces directly to the IOC processor via an 8-bit data bus. The keyboard contains an Intel UPI-41 Universal Peripheral Interface, which scans the keyboard, encodes the characters, and buffers the characters to provide N-key rollover. The keyboard itself is a high quality typewriter-style keyboard containing the full ASCII character set. An upper/lower case switch allows the system to be used for document preparation. Cursor control keys are also provided.

### Floppy Disk Drive

The floppy disk drive is controlled by an Intel 8271 single chip, programmable floppy disk controller. It transfers data via an Intel 8257 DMA controller between an IOC RAM buffer and the diskette. The 8271 handles reading and writing of data, formatting diskettes, and reading status, all upon appropriate commands from the IOC microprocessor.

### Peripheral Interface

A UPI-41 Universal Peripheral Interface on the IOC board performs similar functions to the UPI-41 on the PIO board in the Model 210. It provides interface for other

standard Intellec peripherals, including a printer, high speed paper tape reader, high speed paper tape punch, and universal PROM programmer. Communication between the IPB and IOC is maintained over a separate, 8-bit bidirectional data bus. Connectors for the devices named above, as well as the two serial channels, are mounted directly on the IOC itself.

### Control

User control is maintained through a front panel consisting of a power switch and indicator, reset/boot switch, run/halt light, and eight interrupt switches and indicators. The front-panel circuit board is attached directly to the IPB, allowing the eight interrupt switches to connect to the primary 8259A, as well as to the Intellec Series II bus.

### MULTIBUS Capability

All Intellec Series II models implement the industry-standard MULTIBUS. MULTIBUS enables several bus masters, such as CPU and DMA devices, to share the bus and memory by operating at different priority levels. Resolution of bus exchanges is synchronized by a bus clock signal derived independently from processor clocks. Read/write transfers may take place at rates up to 5 MHz. The bus structure is suitable for use with any Intel microcomputer family.

### Expansion

The Model 220 may be expanded to 64K of RAM and up to 2.25M bytes of on-line diskette storage.

## SPECIFICATIONS

### Host Processor (IPB)

8080A-2 based, operating at 2.600 MHz.

**RAM** — 32K, expandable to 64K with iSBC 032 RAM boards (system monitor occupies 62K through 64K)

**ROM** — 4K (2K in monitor, 2K in boot/diagnostic)

**Bus** — MULTIBUS, maximum transfer rate of 5 MHz

**Clocks** — Host processor, crystal controlled at 2.6 MHz, bus clock, crystal controlled at 9.8304 MHz

### I/O Interfaces

2 Serial I/O Channels, RS232C, at 110-9600 baud (asynchronous) or 150-56K baud (synchronous). Baud rates and serial format fully programmable using Intel 8251A USARTs. Serial Channel 1 additionally provided with 20 mA current loop. Parallel I/O interfaces provided for paper tape punch, paper tape reader, printer, and UPP-103 Universal PROM Programmer.

### Interrupts

8-level, maskable, nested priority interrupt network initiated from front panel or user selected devices.

### Direct Memory Access (DMA)

Standard capability on MULTIBUS; implemented for user selected DMA devices through optional DMA module — maximum transfer rate of 2 MHz.

### Memory Access Time

**RAM** — 585 ns max

**PROM** — 450 ns max

### Diskette

**Diskette System Capacity** — 250K bytes (formatted)

**Diskette System Transfer Rate** — 160K bits/sec

**Diskette System Access Time**

Track-to-Track: 10 ms max

Average Random Positioning: 260 ms max

Rotational Speed: 360 rpm

Average Rotational Latency: 83 ms max

Recording Mode: FM

### Physical Characteristics

**Width** — 17.37 in. (44.12 cm)

**Height** — 15.81 in. (40.16 cm)

**Depth** — 19.13 in. (48.59 cm)

**Weight** — 86 lb (39 kg)

## MODEL 220

### Keyboard

Width — 17.37 in. (44.12 cm)

Height — 3.0 in. (7.62 cm)

Depth — 9.0 in. (22.0 cm)

Weight — 6 lb (3 kg)

### Electrical Characteristics

#### DC Power Supply

Volts Supplied	Amps Supplied	Typical System Requirements
+ 5 ± 5%	30.0	7.5
+ 12 ± 5%	2.5	0.2
- 12 ± 5%	0.3	0.05
- 10 ± 5%	1.5	0.15
+ 15 ± 5%	1.5	1.3*
+ 24 ± 5%	1.7	1.2*

\*Not available on bus.

#### AC Requirements

50-60 Hz. 115/230V AC

### Equipment Supplied

Model 220 chassis

Integrated processor board (IPB)

I/O controller board (IOC)

CRT and keyboard

250K-byte floppy disk drive

ROM resident system monitor

ISIS-II system diskette with MCS-80/MCS-85

macroassembler

### Reference Manuals

**9800558** — A Guide to Microcomputer Development Systems (SUPPLIED)

**9800559** — Intellec Series II Installation and Service Manual (SUPPLIED)

**9800306** — ISIS-II System User's Guide (SUPPLIED)

**9800556** — Intellec Series II Hardware Reference Manual (SUPPLIED)

**9800555** — Intellec Series II Hardware Interface Manual (SUPPLIED)

**9800301** — 8080/8085 Assembly Language Programming Manual (SUPPLIED)

**9800605** — Intellec Series II System Monitor Source Listing (SUPPLIED)

**9800554** — Intellec Series II Schematic Drawing (SUPPLIED)

Reference manuals are shipped with each product only if designated SUPPLIED (see above). Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.

## ORDERING INFORMATION

Part Number	Description
MDS-220	Intellec Series II Model 220 microcomputer development system (110V/60 Hz)
MDS-221	Intellec Series II Model 220 microcomputer development system (220V/50 Hz)





## MODEL 230 INTELLEC® SERIES II MICROCOMPUTER DEVELOPMENT SYSTEM

**Complete microcomputer development center for Intel MCS-86, MCS-80, MCS-85 and MCS-48™ microprocessor families**

**LSI electronics board with CPU, RAM, ROM, I/O, and interrupt circuitry**

**64K bytes RAM memory**

**Self-test diagnostic capability**

**Eight-level nested, maskable priority interrupt system**

**Built-in interfaces for high speed paper tape reader/punch, printer, and universal PROM programmer**

**Integral CRT with detachable upper/lower case typewriter-style full ASCII keyboard**

**Powerful ISIS-II Diskette Operating System software with relocating macroassembler, linker, and locator**

**1 million bytes (expandable to 2.5M bytes) of diskette storage**

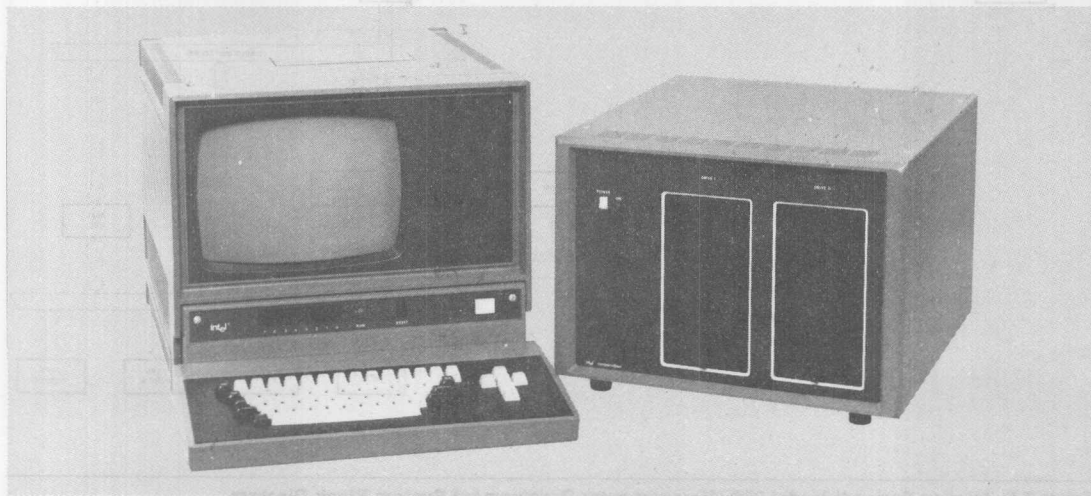
**Supports PL/M and FORTRAN high level languages**

**Standard MULTIBUS with multiprocessor and DMA capability**

**Compatible with standard Intellec®/ISBC**

**Software compatible with previous Intellec® systems**

The Model 230 Intellec Series II Microcomputer Development System is a complete center for the development of microcomputer-based products. It includes a CPU, 64K bytes of RAM, 4K bytes of ROM memory, a 2000-character CRT, a detachable full ASCII keyboard, and dual double density diskette drives providing over 1 million bytes of on-line data storage. Powerful ISIS-II Diskette Operating System software allows the Model 230 to be used quickly and efficiently for assembling and/or compiling and debugging programs for Intel's MCS-86, MCS-80, MCS-85, or MCS-48 microprocessor families without the need for handling paper tape. ISIS-II performs all file handling operations, leaving the user free to concentrate on the details of his own application. When used in conjunction with an optional in-circuit emulator (ICE) module, the Model 230 provides all the hardware and software development tools necessary for the rapid development of a microcomputer-based product.



Intel, Intellec, INSITE, Library Manager, MCS, MEGACHASSIS, MICROAMP, PROMPT,  $\mu$ SCOPE, MULTIBUS, RMX/80, UPI-41, ICE, and ISBC are trademarks of Intel Corp.



## FUNCTIONAL DESCRIPTION

## Hardware Components

The Intellec Series II Model 230 is a packaged, highly integrated microcomputer development system consisting of a CRT chassis with a 6-slot cardcage, power supply, fans, cables, and five printed circuit cards. A separate, full ASCII keyboard is connected with a cable. A second chassis contains two floppy disk drives capable of double-density operation along with a separate power supply, fans, and cables for connection to the main chassis. A block diagram of the Model 230 is shown in Figure 7-5.

**CPU Cards** — The master CPU card contains its own microprocessor, memory, I/O, interrupt and bus interface circuitry fashioned from Intel's high technology LSI components. Known as the integrated processor board (IPB), it occupies the first slot in the cardcage. A second slave CPU card is responsible for all remaining I/O control including the CRT and keyboard interface. This card, mounted on the rear panel, also contains its own microprocessor, RAM and ROM memory, and I/O interface logic, thus, in effect, creating a dual processor environment. Known as the I/O controller (IOC), the slave CPU

card communicates with the IPB over an 8-bit bidirectional data bus.

**Memory and Control Cards** — In addition, 32K bytes of RAM (bringing the total to 64K bytes) is located on a separate card in the main cardcage. Fabricated from Intel's 16K RAMs, the board also contains all necessary address decoding and refresh logic. Two additional boards in the cardcage are used to control the two double-density floppy disk drives.

**Expansion** — Two remaining slots in the cardcage are available for system expansion. Additional expansion of 4 slots can be achieved through the addition of an Intellec Series II expansion chassis.

## System Components

The heart of the IPB is an Intel NMOS 8-bit microprocessor, the 8080A-2, running at 2.6 MHz. 32K bytes of RAM memory are provided on the board using Intel 16K RAMs. 4K of ROM is provided, preprogrammed with system bootstrap "self-test" diagnostics and the Intellec Series II System Monitor. The eight-level vectored priority interrupt system allows interrupts to be individually masked. Using Intel's versatile 8259A interrupt controller, the interrupt system may be user programmed to respond to individual needs.

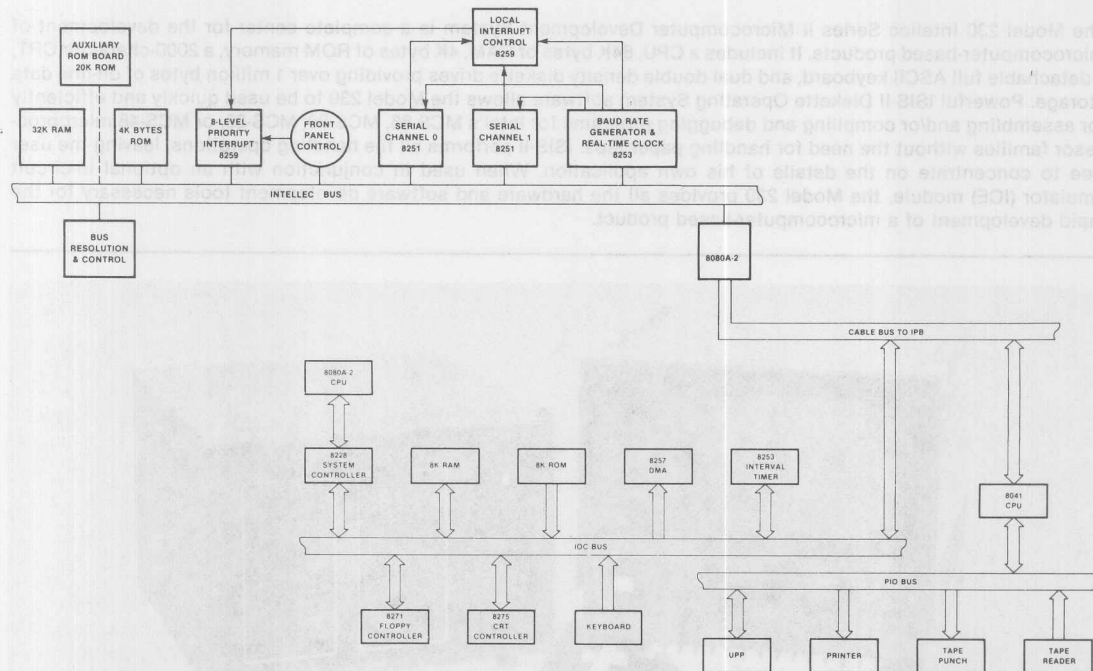


Figure 7-5. Intellec Series II Model 230 Microcomputer Development System Block Diagram

## Input/Output

**IPB Serial Channels** — The I/O subsystem in the Model 230 consists of two parts: the IOC card and two serial channels on the IPB itself. Each serial channel is RS232 compatible and is capable of running asynchronously from 110 to 9600 baud or synchronously from 150 to 56K baud. Both may be connected to a user defined data set or terminal. One channel contains current loop adapters. Both channels are implemented using Intel's 8251A USART. They can be programmatically selected to perform a variety of I/O functions. Baud rate selection is accomplished programmatically through an Intel 8253 interval timer. The 8253 also serves as a real-time clock for the entire system. I/O activity through both serial channels is signaled to the system through a second 8259 interrupt controller, operating in a polled mode nested to the primary 8259.

**IOC Interface** — The remainder of system I/O activity takes place in the IOC. The IOC provides interface for the CRT, keyboard, and standard Intellec peripherals including printer, high speed paper tape reader/punch, and universal PROM programmer. The IOC contains its own independent microprocessor, also an 8080A-2. The CPU controls all I/O operations as well as supervising communications with the IPB. 8K bytes of ROM contain all I/O control firmware. 8K bytes of RAM are used for CRT screen refresh storage. These do not occupy space in Intellec Series II main memory since the IOC is a totally independent microcomputer subsystem.

## Integral CRT

**Display** — The CRT is a 12-inch raster scan type monitor with a 50/60 Hz vertical scan rate and 15.5 kHz horizontal scan rate. Controls are provided for brightness and contrast adjustments. The interface to the CRT is provided through an Intel 8275 single chip programmable CRT controller. The master processor on the IPB transfers a character for display to the IOC, where it is stored in RAM. The CRT controller reads a line at a time into its line buffer through an Intel 8257 DMA controller and then feeds one character at a time to the character generator to produce the video signal. Timing for the CRT control is provided by an Intel 8253 interval timer. The screen display is formatted as 25 rows of 80 characters. The full set of ASCII characters are displayed, including lower case alphas.

**Keyboard** — The keyboard interfaces directly to the IOC processor via an 8-bit data bus. The keyboard contains an Intel UPI-41 Universal Peripheral Interface, which scans the keyboard, encodes the characters, and buffers the characters to provide N-key rollover. The keyboard itself is a high quality typewriter style keyboard containing the full ASCII character set. An upper/lower case switch allows the system to be used for document preparation. Cursor control keys are also provided.

## Peripheral Interface

A UPI-41 Universal Peripheral Interface on the IOC board performs similar functions to the UPI-41 on the PIO board in the Model 210. It provides interface for other standard Intellec peripherals including a printer, high speed paper tape reader, high speed paper tape punch,

and universal PROM programmer. Communication between the IPB and IOC is maintained over a separate 8-bit bidirectional data bus. Connectors for the four devices named above, as well as the two serial channels, are mounted directly on the IOC itself.

## Control

User control is maintained through a front panel, consisting of a power switch and indicator, reset/boot switch, run/halt light, and eight interrupt switches and indicators. The front panel circuit board is attached directly to the IPB, allowing the eight interrupt switches to connect to the primary 8259A, as well as to the Intellec Series II bus.

## Diskette System

The Intellec Series II double density diskette system provides direct access bulk storage, intelligent controller, and two diskette drives. Each drive provides 1/2 million bytes of storage with a data transfer rate of 500,000 bits/second. The controller is implemented with Intel's powerful Series 3000 Bipolar Microcomputer Set. The controller provides an interface to the Intellec Series II system bus, as well as supporting up to four diskette drives. The diskette system records all data in soft sector format. The diskette system is capable of performing seven different operations: recalibrate, seek, format track, write data, write deleted data, read data, and verify CRC.

**Diskette Controller Boards** — The diskette controller consists of two boards, the channel board and the interface board. These two PC boards reside in the Intellec Series II system chassis and constitute the diskette controller. The channel board receives, decodes and responds to channel commands from the 8080A-2 CPU in the Model 230. The interface board provides the diskette controller with a means of communication with the diskette drives and with the Intellec system bus. The interface board validates data during reads using a cyclic redundancy check (CRC) polynomial and generates CRC data during write operations. When the diskette controller requires access to Intellec system memory, the interface board requests and maintains DMA master control of the system bus, and generates the appropriate memory command. The interface board also acknowledges I/O commands as required by the Intellec bus. In addition to supporting a second set of double density drives, the diskette controller may co-reside with the Intel single density controller to allow up to 2.5 million bytes of on-line storage.

## MULTIBUS Capability

All Intellec Series II models implement the industry standard MULTIBUS. MULTIBUS enables several bus masters, such as CPU and DMA devices, to share the bus and memory by operating at different priority levels. Resolution of bus exchanges is synchronized by a bus clock signal derived independently from processor clocks. Read/write transfers may take place at rates up to 5 MHz. The bus structure is suitable for use with any Intel microcomputer family.

## SPECIFICATIONS

### Host Processor (IPB)

**RAM** — 64K (system monitor occupies 62K through 64K)

**ROM** — 4K (2K in monitor, 2K in boot/diagnostic)

### Diskette System Capacity (Basic Two Drives)

#### Unformatted

Per Disk: 6.2 megabits

Per Track: 82.0 kilobits

#### Formatted

Per Disk: 4.1 megabits

Per Track: 53.2 kilobits

### Diskette Performance

**Diskette System Transfer Rate** — 500 kilobits/sec

#### Diskette System Access Time

Track-to-Track: 10 ms

Head Settling Time: 10 ms

**Average Random Positioning Time** — 260 ms

**Rotational Speed** — 360 rpm

**Average Rotational Latency** — 83 ms

**Recording Mode** — M<sup>2</sup>FM

### Physical Characteristics

**Width** — 17.37 in. (44.12 cm)

**Height** — 15.81 in. (40.16 cm)

**Depth** — 19.13 in. (48.59 cm)

**Weight** — 73 lb (33 kg)

#### Keyboard

**Width** — 17.37 in. (44.12 cm)

**Height** — 3.0 in. (7.62 cm)

**Depth** — 9.0 in. (22.86 cm)

**Weight** — 6 lb (3 kg)

#### Dual Drive Chassis

**Width** — 16.88 in. (42.88 cm)

**Height** — 12.08 in. (30.68 cm)

**Depth** — 19.0 in. (48.26 cm)

**Weight** — 64 lb (29 kg)

### Electrical Characteristics

#### DC Power Supply

Volts Supplied	Amps Supplied	Typical System Requirements
+ 5 ± 5%	30	14.25
+ 12 ± 5%	2.5	0.2
- 12 ± 5%	0.3	0.05
- 10 ± 5%	1.5	15
* + 15 ± 5%	1.5	1.3
* + 24 ± 5%	1.7	

\*Not available on bus.

## ORDERING INFORMATION

### Part Number Description

MDS-230	Intellec Series II Model 230 microcomputer development system (110V/60 Hz)
MDS-231	Intellec Series II Model 230 microcomputer development system (220V/50 Hz)

**AC Requirements** — 50/60 Hz, 115/230V AC

### Environmental Characteristics

**Operating Temperature** — 0° to 35°C (95°F)

### Equipment Supplied

Model 230 chassis

Integrated processor board (IPB)

I/O controller board (IOC)

32K RAM board

CRT and keyboard

Double density floppy disk controller (2 boards)

Dual drive floppy disk chassis and cables

2 floppy disk drives (512K byte capacity each)

ROM-resident system monitor

ISIS-II system diskette with MCS-80/MCS-85 macroassembler

### Reference Manuals

**9800558** — A Guide to Microcomputer Development Systems (SUPPLIED)

**9800550** — Intellec Series II Installation and Service Guide (SUPPLIED)

**9800306** — ISIS-II System User's Guide (SUPPLIED)

**9800556** — Intellec Series II Hardware Reference Manual (SUPPLIED)

**9800301** — 8080/8085 Assembly Language Programming Manual (SUPPLIED)

**9800292** — ISIS-II 8080/8085 Assembler Operator's Manual (SUPPLIED)

**9800605** — Intellec Series II Systems Monitor Source Listing (SUPPLIED)

**9800554** — Intellec Series II Schematic Drawings (SUPPLIED)

Reference manuals are shipped with each product only if designated SUPPLIED (see above). Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.



## ISIS-II DISKETTE OPERATING SYSTEM MICROCOMPUTER DEVELOPMENT SYSTEM

**Supports up to two hard disk drives (4 platters), four double density drives and two single density drives, providing up to 17 megabytes of storage in one system with up to 200 files per diskette, and 992 files per disk platter**

**Relocating MCS-80/MCS-85™ macro-assembler contains extended macro and conditional assembly capability**

**Command file facility allows console commands to be submitted from disk file**

**Disk operating system functions callable from user programs**

**Disk system text editor provides string search, substitution, insertions, and deletion commands**

**Supports resident, high level programming languages, PL/M, FORTRAN, BASIC, and COBOL**

**Provides dynamic allocation and deallocation of disk sectors for variable length files**

**Linker automatically combines separately assembled or compiled programs into single relocatable module**

**Library manager™ creates and updates program libraries**

**Supports all standard Intellec® peripherals**

**Provides access to all Intellec® monitor facilities**

The ISIS-II Microcomputer Development System Disk Operating System is a sophisticated, general purpose, high speed data handler and file manipulation system. It provides the ability to edit, assemble, compile, link, relocate, execute, and debug programs, and performs all user file management tasks. The ISIS-II operating system resides on the system disk and supports a broad range of user oriented design aid software. Total file management and input editing features greatly reduce software development time. The ISIS-II relocating macroassembler, linker, object locator, and library manager may be loaded from the disk in seconds. All passes of the assembler may be executed without the need for user intervention. Object code and listings may be directed to any output device, or stored as disk files. Powerful system console commands are provided in an easy to use context. Monitor mode may be entered by a special prefix to any system command or program call.





and supports a broad range of user oriented design aid software. Total file management and input editing features greatly reduce software development time. The ISIS-II relocating macroassembler, linker, object locator, and library manager may be loaded from the disk in seconds. All passes of the assembler may be executed without the need for user intervention. Object code and listings may be directed to any output device, or stored as disk files. A diagram of the ISIS-II system program development flow is shown in Figure 7-6.

### ISIS-II Files

A file is a user defined collection of information of variable length. ISIS-II also treats each of the standard Intellec system peripherals as files through preassignment of unique file names to each device. In this manner data may be copied from one device to another (i.e., tape reader to tape punch) using the same command required to copy one disk data file to another. ISIS-II provides automatic implementation of random access disk files. Each file is identified by a user chosen name unique on its disk. Up to 200 files may be stored on each disk, 992 on each hard disk platter.

### ISIS-II System Commands

ISIS-II system commands are designed to provide the user with a powerful, easy to use program and file manipulation capability. Several commands have the capability of operating on several files at once via the wildcard file naming convention. As an example, the command DELETE \* .OBJ deletes all files in the disk directory with the suffix .OBJ. A summary of ISIS-II system commands is presented in Table 7-4.

allows the user to open, close, read, and write disk files, access standard peripheral devices, write error messages, and load other programs via simple program call statements.

Command	Operation
Initialize disk	Initializes a diskette for use by the system. Requires only one disk drive.
Attribute assignment	Assigns specified attributes to a file, such as write-protect.
Copy	Creates copies of existing diskette files or transfers files from one device to another.
Delete	Removes a file from the diskette, thereby freeing space for allocation of other files.
Directory	Lists name, size, and attributes of files from a specified diskette directory.
Rename	Allows diskette files to be renamed.
Format	Initializes a diskette for use by the system. (Use with two or more drives.)
Debug	Loads a specified program from a diskette into memory and then transfers control to the Intellec monitor for execution and or debugging.
Submit	Provides capability for executing a series of ISIS-II commands previously written to a diskette file.

Additional commands are provided for support of the hard disk.

Table 7-4. ISIS-II System Commands

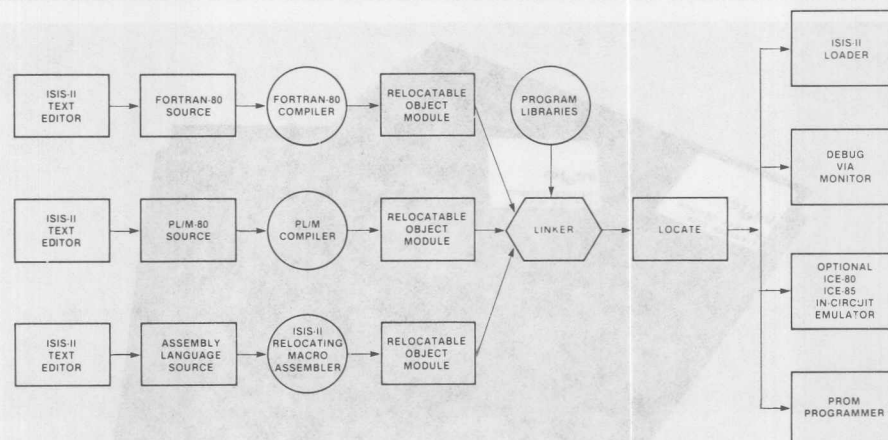


Figure 7-6. Program Development Flow Using ISIS-II Disk Operating System



## ISIS-II Text Editor

The ISIS-II text editor is a comprehensive tool for assembly language, PL/M, and FORTRAN program entry and correction for Intel microcomputers. Its command set allows either entire lines of text or individual characters to be manipulated within a line.

**Program Entry** — Programs may be entered from the console keyboard or may be loaded directly. Text is stored internally in the editor's workspace, and may be edited with the following commands:

- string insertion or deletion
- string search
- string substitution

**Utility Commands** — To facilitate the use of these editing commands, utility commands are used to change positions in the workspace. These include:

- move pointer by line or by character
- move pointer to start of workspace
- move pointer to end of workspace

**Storage** — The contents of the workspace are stored on diskette and can be immediately accessed by ISIS-II commands or other programs, such as the ISIS-II MCS-80/MCS-85 macroassembler.

For users with 64K of RAM memory, the CREDIT Text Editor is available. See CREDIT data sheet for more information.

## ISIS-II MCS-80/MCS-85™ Relocating Macroassembler

**Address Translation** — The ISIS-II MCS-80/MCS-85 macroassembler translates assembly language mnemonics into relocatable and/or absolute object code modules. In addition to eliminating the errors of hand translation, the ability to refer to program addresses with symbolic names makes it easy to modify programs by adding or deleting instructions. Extended macro capability eliminates the need to rewrite similar sections of code repeatedly, and thus simplifies program documentation. Conditional assembly permits the assembler to include or delete sections of code that may vary from system to system, such as the code required to handle optional external devices. Additionally, the user is allowed complete freedom in assigning the location of code, data, and stack segments.

**List File** — The ISIS-II Assembler accepts disk file input and produces a relocatable object file with corresponding symbol table and assembly listing file, including any

error messages. A cross reference listing is also optionally produced. The list file may then be examined from the system console or copied to a specified list device.

**Object File** — The relocatable object file generated by the assembler may be combined with other object programs residing on the disk to form a single relocatable object module or it can be converted to an absolute form for subsequent loading and execution.

## ISIS-II Linker

The ISIS-II linker provides the capability to combine the outputs of several independently compiled or assembled object modules (files) into a single relocatable object module. The linker automatically resolves all external program and data references during the linking process. Object modules produced from previous link operations may be easily linked to a new module. ISIS-II also provides facilities to ease the generation of overlays. An optional link map showing the contents and lengths of each segment in the output module can be requested. All unsatisfied external references are also listed. If requested by the user, the ISIS-II linker can search a specified set of program libraries for routines to be included in the output module.

## ISIS-II Object Locator

The ISIS-II locate program takes output from either the resident FORTRAN or PL/M compilers, the macroassembler, or the linker and transforms that output from relocatable format to an absolute format which may then be loaded via the standard ISIS-II loader, or loaded into an appropriate in-circuit emulator (ICE) module. During the locate process, code, data, and stack segments may be separately relocated, allowing code to be put in areas to be subsequently specified as ROM, while data and the stack are directed to RAM addresses. A locate map showing absolute addresses for each code and data segment and a symbol table dump listing symbols, attributes, and absolute address may also be requested.

## ISIS-II Library Manager™

The ISIS-II Library Manager program provides for the creation and maintenance of a program library containing Intel-provided and user-written programs and sub-routines. These library routines may be linked to a program using the ISIS-II linker. Several libraries, each containing its own set of routines, may be created.

## ORDERING INFORMATION

### Part Number Description

Included with all Intellec Series II Microcomputer Development Systems.

Not available separately.



## PL/M-80

# HIGH LEVEL PROGRAMMING LANGUAGE INTELLEC® RESIDENT COMPILER

**Provides resident operation on Intellec® Microcomputer Development System and Intellec® Series II microcomputer development systems**

**Produces relocatable and linkable object code**

**Sophisticated code optimization reduces application memory requirements**

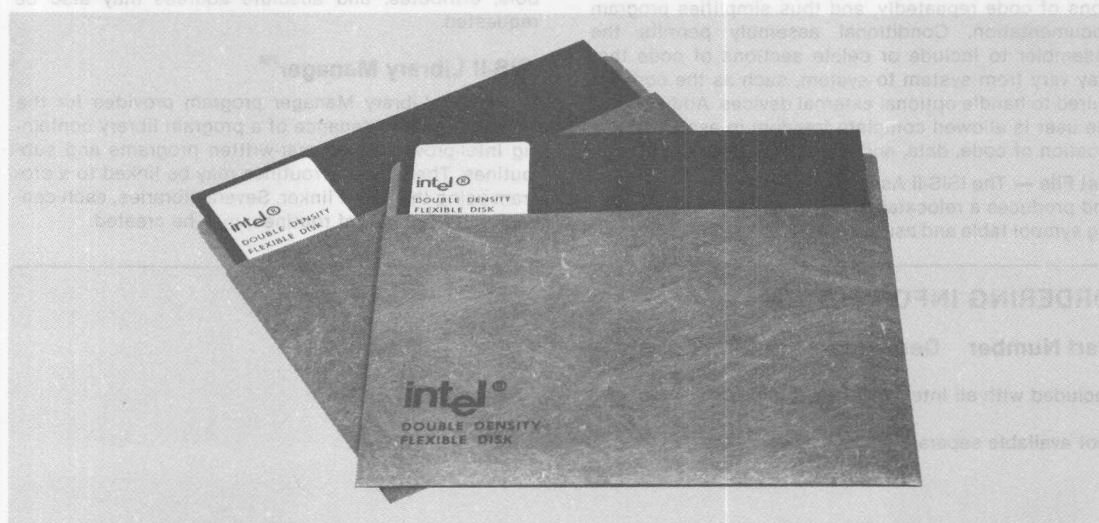
**Speeds project completion with increased programmer productivity**

**Cuts software development and maintenance costs**

**Improves product reliability with simplified language and consequent error reduction**

**Eases enhancement as system capabilities expand**

The PL/M-80 High Level Programming Language Intellec Resident Compiler is an advanced, high level programming language for Intel 8080 and 8085 microprocessors, iSBC-80 OEM computer systems, and Intellec microcomputer development systems. PL/M has been substantially enhanced since its introduction in 1973 and has become one of the most effective and powerful microprocessor systems implementation tools available. It is easy to learn, facilitates rapid program development and debugging, and significantly reduces maintenance costs. PL/M is an algorithmic language in which program statements naturally express the algorithm to be programmed, thus freeing programmers to concentrate on system development rather than assembly language details (such as register allocation, meanings of assembler mnemonics, etc.). The PL/M compiler efficiently converts free-form PL/M programs into equivalent 8080/8085 instructions. Substantially fewer PL/M statements are necessary for a given application than would be using assembly language or machine code. Since PL/M programs are problem oriented and thus more compact, programming in PL/M results in a high degree of productivity during development efforts, resulting in significant cost reduction in software development and maintenance for the user.



## FUNCTIONAL DESCRIPTION

The PL/M compiler is an efficient multiphase compiler that accepts source programs, translates them into object code, and produces requested listings. After compilation, the object program may be first linked to other modules, then located to a specific area of memory, and finally executed. The diagram shown in Figure 7-7 illustrates a program development cycle where the program consists of three modules: PL/M, FORTRAN, and assembly language. A typical PL/M compiler procedure is shown on the following page.

### Features

Major features of the Intel PL/M-80 compiler and programming language include:

**Resident Operation** — on Intel microcomputer development systems eliminates the need for a large in-house computer or costly timesharing system.

**Object Code Generation** — of relocatable and linkable object codes permits PL/M program development and debugging in small modules, which may be easily linked with other modules and/or library routines to form a complete application.

**Extensive Code Optimization** — including compile time arithmetic, constant subscript resolution, and common subexpression elimination, results in generation of short, efficient CPU instruction sequences.

**Symbolic Debugging** — fully supported in the PL/M compiler and ICE-85 in-circuit emulators.

**Compile Time Options** — includes general listing format commands, symbol table listing, cross reference listing, and "innerlist" of generated assembly language instructions.

**Block Structure** — aids in utilization of structured programming techniques.

**Access** — provided by high level PL/M statements to hardware resources (interrupt systems, absolute addresses, CPU input/output ports).

**Data Definition** — enables complex data structures to be defined at a high level.

**Re-entrant Procedures** — may be specified as a user option.

### Benefits

PL/M is designed to be an efficient, cost-effective solution to the special requirements of microcomputer software development as illustrated by the following benefits of PL/M use:

**Low Learning Effort** — even for the novice programmer, because PL/M is easy to learn.

**Earlier Project Completion** — on critical projects, because PL/M substantially increases programmer productivity while reducing program development time.

**Lower Development Cost** — because increased programmer productivity requiring less programming resources for a given function translates into lower software development costs.

**Increased Reliability** — because of PL/M's use of simple statements in the program algorithm, which are easier to correct and thus substantially reduce the risk of costly errors in systems that have already reached full production status.

**Easier Enhancement and Maintenance** — because programs written in PL/M are easier to read and easier to understand than assembly language, and thus are easier to enhance and maintain as system capabilities expand and future products are developed.

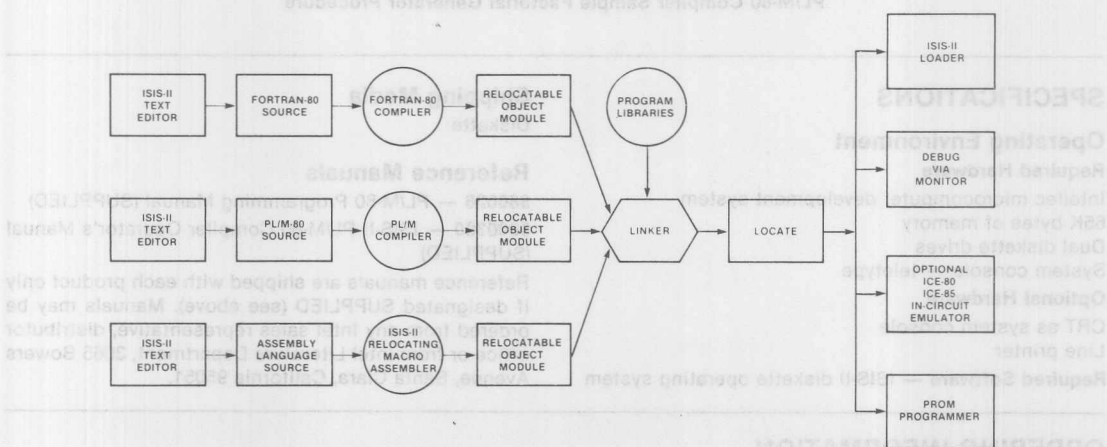


Figure 7-7. Program Development Cycle Block Diagram

**Simpler Project Development** — because the Intel microcomputer development system with resident PL/M-80 is all that is needed for developing and debug-

ging software for 8080 and 8085 microcomputers, and the use of expensive (and remote) timesharing or large computers is consequently not required.

		\$OBJECT(F1:FACT.OB2)
		\$DEBUG
		\$XREF
		\$TITLE('FACTORIAL GENERATOR — PROCEDURE')
		\$PAGEWIDTH(80)
1		FACT:
		DO;
2	1	DECLARE NUMCH BYTE PUBLIC;
3	1	FACTORIAL: PROCEDURE (NUM,PTR) PUBLIC;
4	2	DECLARE NUM BYTE, PTR ADDRESS;
5	2	DECLARE DIGITS BASED PTR (161) BYTE;
6	2	DECLARE (I,C,M) BYTE;
7	2	NUMCH = 1; DIGITS(1) = 1;
9	2	DO M = 1 TO NUM;
10	3	C = 0;
11	3	DO I = 1 TO NUMCH;
12	4	DIGITS(I) = DIGITS(I)*M + C;
13	4	C = DIGITS(I)/10;
14	4	DIGITS(I) = DIGITS(I) — 10*C;
15	4	END;
16	3	IF C<>0 THEN
17	3	DO;
18	4	NUMCH = NUMCH + 1; DIGITS(NUMCH) = C;
20	4	C = DIGITS(NUMCH)/10;
21	4	DIGITS(NUMCH) = DIGITS(NUMCH) — 10*C;
22	4	END
		END;
24	2	END FACTORIAL;
25	1	END;

PL/M-80 Compiler Sample Factorial Generator Procedure

## SPECIFICATIONS

### Operating Environment

#### Required Hardware

Intel microcomputer development system  
65K bytes of memory  
Dual diskette drives  
System console — teletype

#### Optional Hardware

CRT as system console  
Line printer

**Required Software** — ISIS-II diskette operating system

### Shipping Media

Diskette

### Reference Manuals

**980026** — PL/M-80 Programming Manual (SUPPLIED)

**9800300** — ISIS-II PL/M-80 Compiler Operator's Manual (SUPPLIED)

Reference manuals are shipped with each product only if designated SUPPLIED (see above). Manuals may be ordered from any Intel sales representative, distributor office or from Intel Literature Department, 3065 Bowers Avenue, Santa Clara, California 95051.

## ORDERING INFORMATION

### Product Code Description

MDS-PLM High level language compiler





## ICE-85™ MCS-85™ IN-CIRCUIT EMULATOR

**Connects the Intellec® System Resources to the user-configured system via a 40-pin adaptor plug**

**Executes user system software in real-time**

**Allows user-configured system to share Intellec® memory and I/O facilities**

**Provides 1023 states of 8085 trace data plus 18 additional logic signals via an External Trace Module**

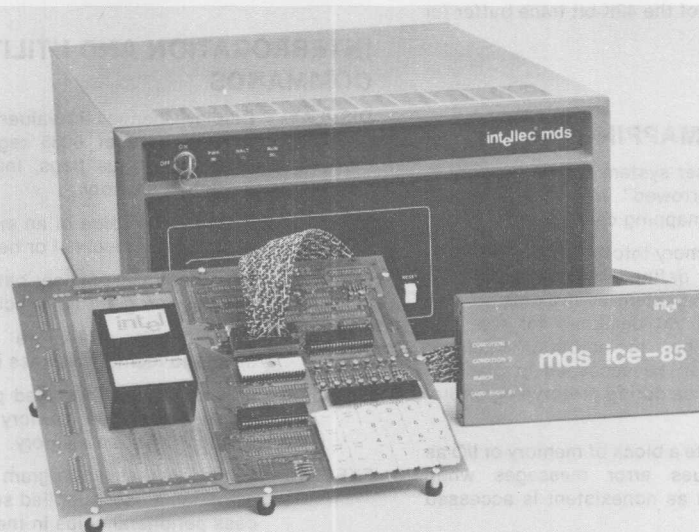
**Offers full symbolic debugging capability for both assembly language and Intel's high-level compiler languages PL/M-80 and FORTRAN-80**

**Displays trace data from the user's 8085 in assembler mnemonics and allows personality groupings of data sampled by the external 18-channel trace module**

**Extends ICE capabilities to the rest of the prototype system peripheral circuitry by allowing the user to execute his own peripheral chip analysis routines**

**Provides ability to examine and alter MCS-85™ registers, memory, flag values, interrupt bits and I/O ports**

The ICE-85 module resides in the Intellec® Microcomputer Development System and interfaces to the user system's 8085. In addition, an external trace module provides access to user system peripheral circuitry via a user-configured DIP clip for peripheral ICs or may be attached to as many as 18 separate prototype signal nodes via individual probe clips. Using the ICE-85 module, the designer can execute prototype software in real-time or single-step mode and can substitute Intellec® system memory and I/O for user system equivalent. ICE capability can be extended to the rest of the user system peripheral circuitry by allowing the user to create and execute a library of user-defined peripheral chip analyzer routines. All user access to the prototype system software may be done symbolically by assigning names to program locations and data, I/O ports and groups of external trace signals. For the first time, in-circuit emulation extends beyond the user's prototype CPU to the entire user's system, allowing In-System Emulation.





## SYMBOLIC DEBUGGING CAPABILITY

ICE-85 allows the user to make symbolic references to I/O ports, memory addresses and data in his program. Symbols and PL/M-80 statement number may be substituted for numeric values in any of the ICE-85 commands. The user is relieved from looking up addresses of variables or program subroutines.

The user symbol table generated along with the object file during a PL/M-80 or FORTRAN-80 compilation or by the ISIS-II 8080/8085 Macro Assembler is loaded into the Intellec® System memory along with the user program which is to be emulated. The user may add to this symbol table any additional symbolic values for memory addresses, constants, or variables that are found useful during system debugging. By referring to symbol memory addresses, the user can examine, change or break at the intended location.

ICE-85 provides symbolic definition of all 8085 registers, interrupt bits and flags. The following symbolic references are also provided for user convenience: TIMER, the low-order 16 bits of a register containing the number of 2 MHz clock pulses elapsed during emulation; HTIMER, the high-order 16 bits of the timer counter; PPC, the address of the last instruction emulated; BUFFERSIZE, the number of frames of valid trace data (between 0 and 1022).

## PERSONALITY GROUPED DISPLAYS

Trace data in the 1023 by 42-channel real-time trace memory buffer is displayed in easy to read format. The user has the option to specify trace data displays in actual 8085 assembler instruction mnemonics. The data collected from the External Trace Module can be grouped and symbolically named according to user specifications and displayed in the appropriate number base designation. Simple ICE-85 commands allow the user to select any portion of the 42K-bit trace buffer for immediate display.

## MEMORY AND I/O MAPPING

Memory and I/O for the user system can be resident in the user system or "borrowed" from the Intellec® System through ICE-85's mapping capability.

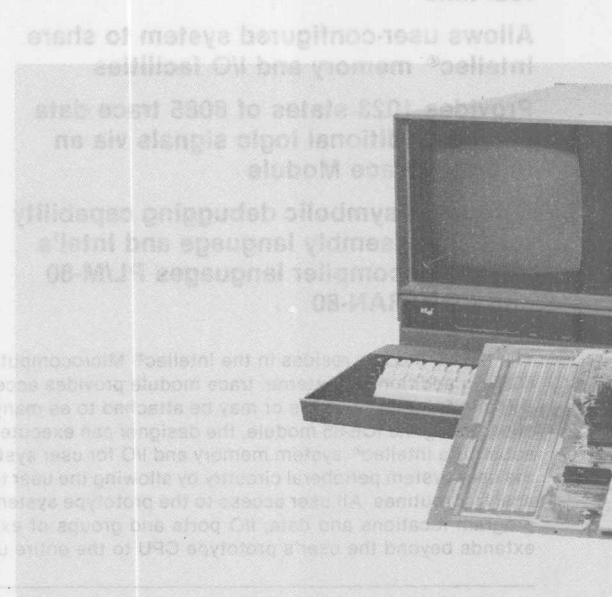
ICE-85 separates user memory into 32 2K blocks. Each block of memory can be defined independently. The user may assign Intellec® System equivalents to take the place of devices not yet designed for the user system during prototyping. In addition, Intellec® System memory or I/O can be accessed in place of suspect user system devices during prototyping or production checkout.

The user can also designate a block of memory or I/O as nonexistent. ICE-85 issues error messages when memory or I/O designated as nonexistent is accessed by the user program.

## INTEGRATED HARDWARE/SOFTWARE DEVELOPMENT

The user prototype need consist of no more than an 8085 CPU socket and a user bus to begin integration of software and hardware development efforts. Through ICE-85 mapping capabilities, Intellec® System equivalents can be accessed for missing prototype hardware. Hardware designs can be tested using the system software which will drive the final product.

The system integration phase, which can be so costly when attempting to mesh completed hardware and software products, becomes a convenient two-way debug tool when begun early in the design cycle.



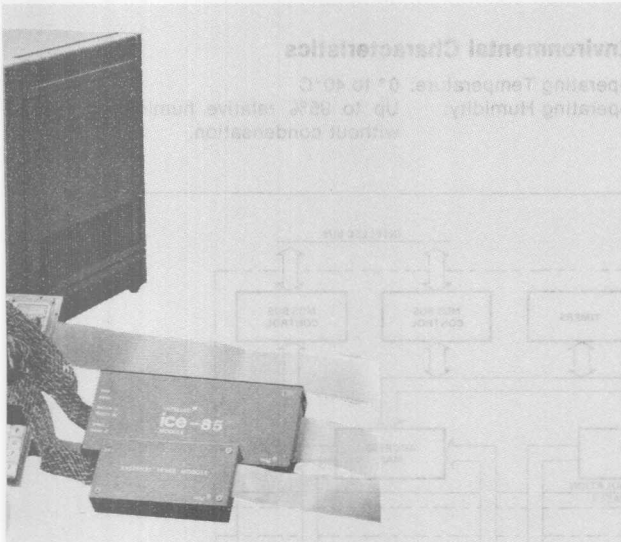
## INTERROGATION AND UTILITY COMMANDS

DISPLAY/CHANGE	Display/Changes the values of symbols and the contents of 8085 registers, pseudo-registers, status flags, interrupt bits, I/O ports and memory.
EVALUATE	Displays the value of an expression in the binary, octal, decimal or hexadecimal.
SEARCH	Searches user memory between locations in a user program for specified contents.
CALL	<i>Emulates</i> a procedure starting at a specified memory address in user memory.
ICALL	<i>Executes</i> a user-supplied procedure starting at a specified memory address in the Intellec® System memory.
EXECUTE	Saves emulated program registers and emulates a user-supplied subroutine to access peripheral chips in the user's system.

## REAL TIME TRACE

ICE-85 captures valuable trace information from the emulating CPU and the External Trace Module while the user is executing programs in real time. The 8085 status, the user memory or port addressed, the data read or written, the serial data lines and data from 18 external signals, is stored for the last 1023 machine states executed (511 machine cycles). This provides ample data for determining how the user system was reacting prior to emulation break. It is available whether the break was user-initiated or the result of an error condition.

For detailed information on the actions of CPU registers, flags, or other system operations, the user may operate in single or multi-step sequences tailored to system debug needs.



## EMULATION CONTROLS AND COMMANDS

- |              |   |
|--------------|---|
| <b>GROUP</b> | Defines into a symbolically named group, a channel or combination of channels from the 8085 Microprocessor and/or the External Trace Module.  |
| <b>GO</b>    | Initiates real-time emulation and controls emulation break conditions.  |
| <b>STEP</b>  | Initiates emulation in single instruction steps. User may specify the type and amount of information displayed following each step, and define conditions under which stepping should continue. |
| <b>PRINT</b> | Prints the user-specified portion of the trace memory to the selected list device.  |

## EXTERNAL TRACE MODULE

TTL level signals from 18 points in the user system may be synchronously sampled by the External Trace Module and collected in ICE-85's trace buffer. The signals can be collected from a single peripheral chip via the supplied 40-pin DIP clip or may be placed by the user on up to 18 separate signal nodes using the supplied 18 individual probe clips. These signals are included in the 42-channel breakpoint comparisons and clock qualifiers. Also, data from these 18 channels may be displayed in each to read, user-defined groupings.

## SYNCHRONOUS OPERATION WITH OTHER DESIGN AIDS

ICE-85 can be synchronized with other Intellect® design aids by means of two external synchronization lines. These lines are used to enable and disable ICE-85 trace data collection and to cause break conditions based on an external signal which may not be included in the ICE-85 breakpoint registers. In addition, ICE-85 can generate signals on these lines which may be used to control other design aids.

## BREAK REGISTERS/TRACE MEMORY

ICE-85 has two breakpoint registers which are used to break emulation, and two trace qualifier registers which are used to control the collection of trace data during emulation. Each register is 42 entries wide, one entry for each channel and each entry can take any one of the three values 0, 1 or "don't care."

The trace buffer, also 42 entries wide, collects data sampled from 24 8085 processor channels and 18 external channels sampled by the External Trace Module. The signals collected from the 8085 include address lines, data lines, status lines and series input and output lines. The 18 channels extending from the External Trace Module synchronously sample and collect into the trace buffer any user-specified TTL compatible signal from the rest of the prototype system. "Break" and "trace qualification" may therefore occur as a result of a match of any combination of up to 42 channels of CPU and external circuitry signals.

## ICE-85™ IN-CIRCUIT EMULATOR

### Required Hardware:

Intellec® Microcomputer Development System  
System Console  
Intellec® Diskette Operating System  
ICE-85 Module  
Required Software:  
System Monitor  
ISIS-II  
ICE-85 Software

Height: 6.75 in. (17.13 cm)  
Depth: 0.50 in. (1.27 cm)  
Packaged Weight: 6.00 lb (2.73 kg)

### Electrical Characteristics

#### DC Power:

$V_{CC} = +5V \pm 5\%$   
 $I_{CC} = 12A$  maximum; 10A typical  
 $V_{DD} = +12V \pm 5\%$   
 $I_{DD} = 80$  mA maximum; 60 mA typical  
 $V_{BB} = -10V \pm 5\%$   
 $I_{BB} = 1$  mA maximum; 10  $\mu A$  typical

### Equipment Supplied

18-Channel External Trace Module  
Printed Circuit Boards (2)  
Interface Cable and Emulation Buffer Module  
Operator's Manual  
ICE-85 Software, Diskette-Based Version

### Emulation Clock

User's system clock or ICE-85 adaptor socket  
(6.144 MHz Crystal)

### Environmental Characteristics

Operating Temperature: 0° to 40°C  
Operating Humidity: Up to 95% relative humidity  
without condensation.

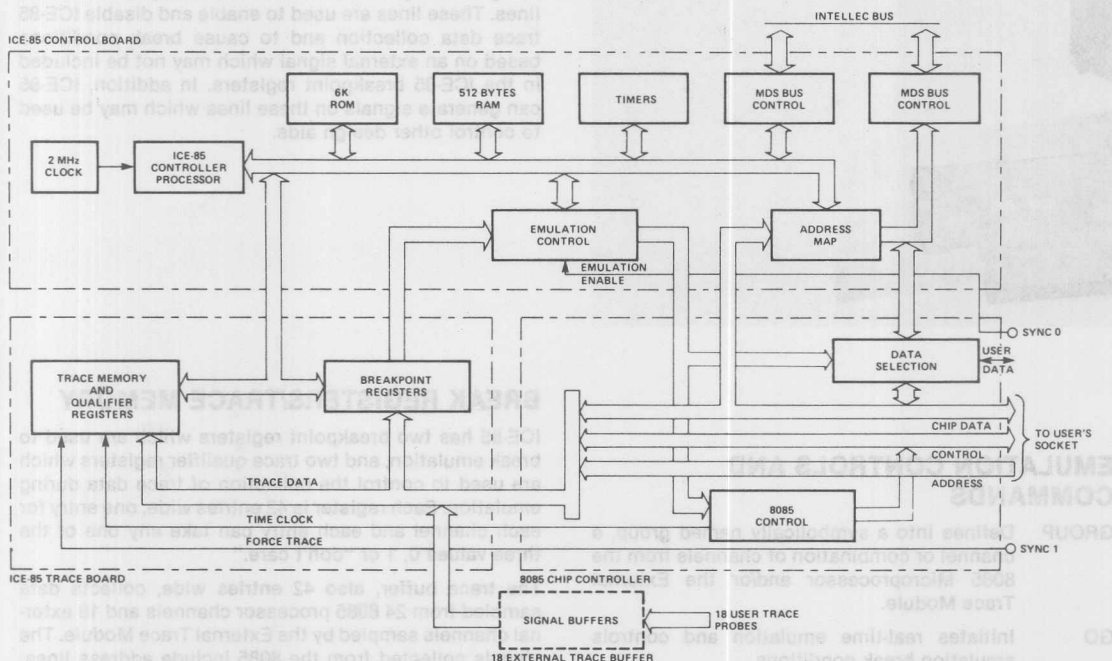


Figure 7-8. ICE-85 Block Diagram

### Ordering Information

#### Part Number Description

MDS-85-ICE 8085 CPU In-Circuit Emulator and  
18-Channel External Trace Module



SDK-85

## SDK-85 MCS-85™ SYSTEM DESIGN KIT

**Complete single board microcomputer system including CPU, memory, and I/O**

**Easy to assemble, low cost, kit form**

**Extensive system monitor software in ROM**

**Interactive LED display and keyboard**

The SDK-85 MCS-85 System Design Kit is a complete single board microcomputer system in kit form. It contains all components required to complete construction of the kit, including LED display, keyboard, resistors, caps, crystal, and miscellaneous hardware. Included is a preprogrammed ROM containing a system monitor for general software utilities and system diagnostics. The complete kit includes a 6-digit LED display and a 24-key keyboard for a direct insertion, examination, and execution of a user's program. In addition, it can be directly interfaced with a teletype terminal. The SDK-85 is an inexpensive, high performance prototype system that has designed-in flexibility for simple interface to the user's application.

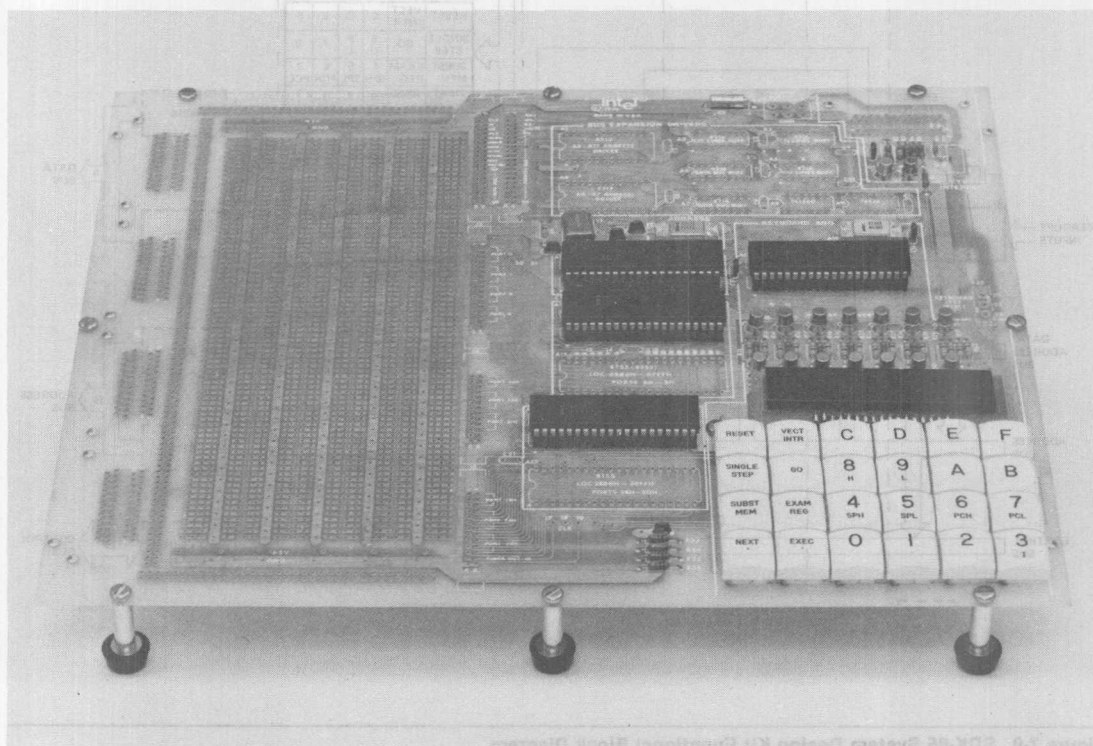
**Large wire-wrap area for custom interfaces**

**Popular 8080A instruction set**

**Interfaces directly with TTY**

**High performance 3 MHz 8085A CPU  
(1.3  $\mu$ s instruction cycle)**

**Comprehensive design library included**





system  
essary  
Such

functional block  
acetylcholine

CPU		ADDRESS DECODER	ROM/IO (8355) EPROM/IO (8755)	RAM/IO/COUNTER	KEYBOARD/DISPLAY		FOR BUS EXPANSION	
ADDRESS	DATA				ADDRESS	DATA		

**Addressing** — The 8085A uses a multiplexed data bus.

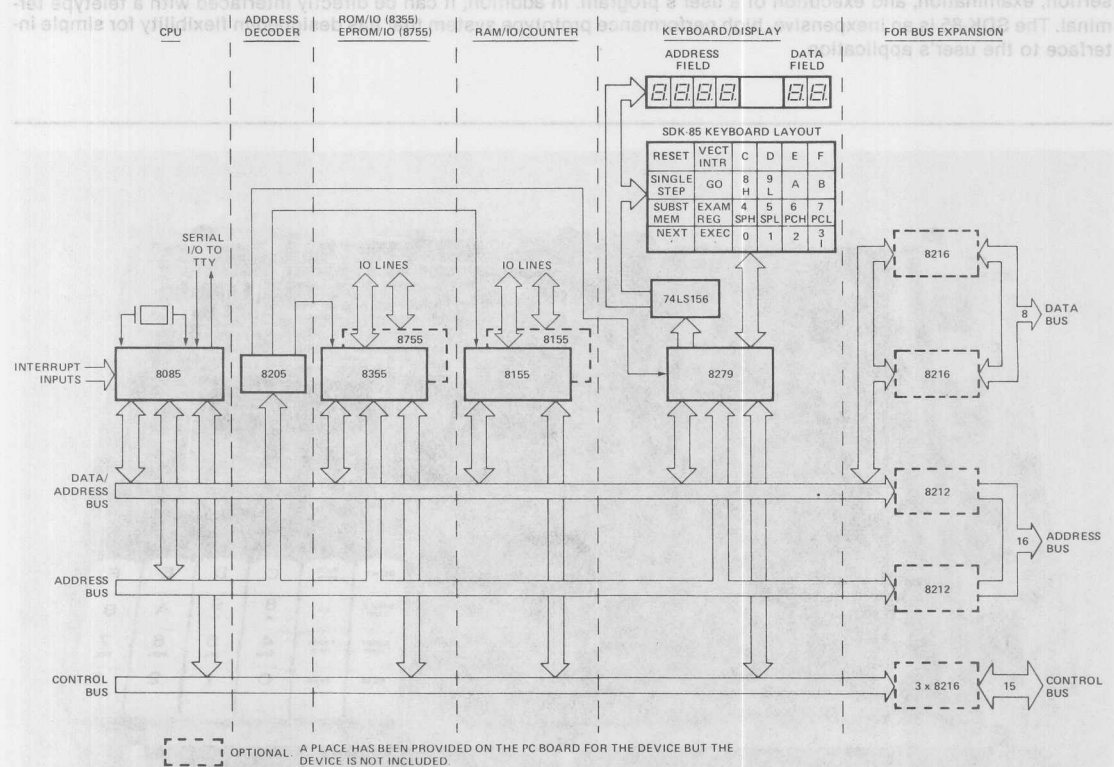
### System Monitor

ROM.

### Communications Interface

The GDK 85 communicates with

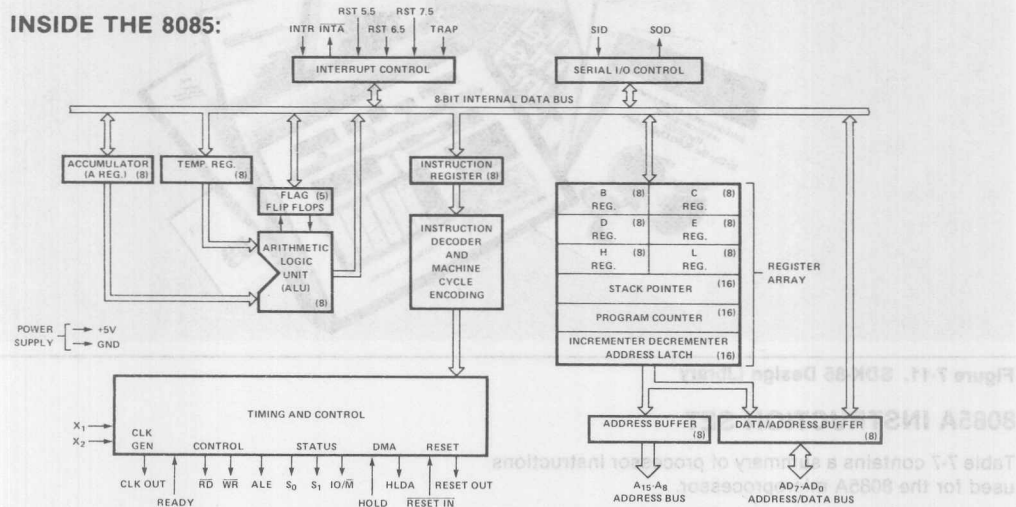
The SDK-65 MCS-65 System Design Kit is a complete and



**Figure 7-9. SDK-85 System Design Kit Functional Block Diagram**



# INSIDE THE 8085:



- SEVEN 8-BIT REGISTERS. SIX OF THEM CAN BE LINKED IN REGISTER PAIRS FOR CERTAIN OPERATIONS.
- 8-BIT ALU.

- 16-BIT STACK POINTER (STACK IS MAINTAINED OFFBOARD IN SYSTEM RAM MEMORY).
- 16-BIT PROGRAM COUNTER.

Figure 7-10. 8085A Microprocessor Block Diagram

Both memory and I/O can be easily expanded by simply soldering in additional devices in locations provided for this purpose. A large area of the board (45 sq. in.) is laid out as general purpose wire-wrap for the user's custom interfaces.

## Assembly

Only a few simple tools are required for assembly; soldering iron, cutters, screwdriver, etc. The SDK-85 user's manual contains step-by-step instructions for easy assembly without mistakes. Once construction is complete, the user connects his kit to a power supply and the SDK-85 is ready to go. The monitor starts immediately upon power-on or reset.

Command	Operation
Reset	Starts monitor.
Go	Allows user to execute user program.
Single step	Allows user to execute user program one instruction at a time—useful for debugging.
Substitute memory	Allows user to examine and modify memory locations.
Examine register	Allows user to examine and modify 8085A's register contents.
Vector interrupt	Serves as user interrupt button.

Table 7-5. Keyboard Monitor Commands

**Commands** — Keyboard monitor commands and teletype monitor commands are provided in Table 7-5 and Table 7-6, respectively.

Command	Operation
Display memory	Displays multiple memory locations.
Substitute memory	Allows user to examine and modify memory locations one at a time.
Insert instructions	Allows user to store multiple bytes in memory.
Move memory	Allows user to move blocks of data in memory.
Examine register	Allows user to examine and modify the 8085A's register contents.
Go	Allows user to execute user programs.

Table 7-6. Teletype Monitor Commands

## Documentation

In addition to detailed information on using the monitors, the SDK-85 user's manual provides circuit diagrams, a monitor listing, and a description of how the system works. The complete design library for the SDK-85 is shown in Figure 7-11 and listed in the Specifications section under Reference Manuals.



Figure 7-11. SDK-85 Design Library

## 8085A INSTRUCTION SET

Table 7-7 contains a summary of processor instructions used for the 8085A microprocessor.

Mnemonic <sup>1</sup>	Description	Instruction Code <sup>2</sup>								Clock <sup>3</sup> Cycles
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	
MOVE, LOAD, AND STORE										
MOV r1r2	Move register to register	0	1	D	D	D	S	S	S	4
MOV M.r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r.M	Move memory to register	0	1	D	D	D	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
XCHG	Exchange D & E, H & L registers	1	1	1	0	1	0	1	1	4
STACK OPS										
PUSH B	Push register pair B & C on stack	1	1	0	0	0	0	1	0	12
PUSH D	Push register pair D & E on stack	1	1	0	1	0	1	0	1	12
PUSH H	Push register pair H & L on stack	1	1	1	0	0	1	0	1	12
PUSH PSW	Push A and flags on stack	1	1	1	1	0	1	0	1	12
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and flags off stack	1	1	1	1	0	0	0	1	10
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	16
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	6
Arithmetic and Logic Instructions										
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	6
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	6
JUMP										
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	7/10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	7/10
JZ	Jump on zero	1	1	0	0	1	0	1	0	7/10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	7/10
JP	Jump on positive	1	1	1	1	0	0	1	0	7/10
JM	Jump on minus	1	1	1	1	1	0	1	0	7/10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	7/10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	7/10
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	6
CALL										
CALL	Call unconditional	1	1	0	0	1	1	0	1	18
CC	Call on carry	1	1	0	1	1	1	0	0	9/18
CNC	Call on no carry	1	1	0	1	0	1	0	0	9/18
CZ	Call on zero	1	1	0	0	1	1	0	0	9/18
CNZ	Call on no zero	1	1	0	0	0	1	0	0	9/18
CP	Call on positive	1	1	1	1	0	1	0	0	9/18
CM	Call on minus	1	1	1	1	1	1	0	0	9/18
CPE	Call on parity even	1	1	1	0	1	1	0	0	9/18
CPO	Call on parity odd	1	1	1	0	0	1	0	0	9/18
RETURN										
RET	Return	1	1	1	0	0	1	0	0	10
RC	Return on carry	1	1	1	0	1	1	0	0	6/12
RNC	Return on no carry	1	1	1	0	0	1	0	0	6/12
RZ	Return on zero	1	1	1	0	0	1	0	0	6/12
RNZ	Return on no zero	1	1	1	0	0	0	0	0	6/12
RP	Return on positive	1	1	1	1	0	0	0	0	6/12
RM	Return on minus	1	1	1	1	1	0	0	0	6/12

continued

Mnemonic <sup>1</sup>	Description	Instruction Code <sup>2</sup>								Clock <sup>3</sup> Cycles
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	
RPE	Return on parity even	1	1	1	0	1	0	0	0	6/12
RPO	Return on parity odd	1	1	1	0	0	0	0	0	6/12
<b>RESTART</b>										
RST	Restart	1	1	1	A	A	A	1	1	12
<b>INCREMENT AND DECREMENT</b>										
INR r	Increment register	0	0	D	D	D	1	0	0	4
DCR r	Decrement register	0	0	D	D	D	1	0	1	4
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	6
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	6
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	6
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	6
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	6
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	6
<b>ADD</b>										
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	0	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
<b>SUBTRACT</b>										
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
<b>LOGICAL</b>										
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
<b>ROTATE</b>										
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
<b>SPECIALS</b>										
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
<b>INPUT/OUTPUT</b>										
IN	Input	1	1	0	1	0	1	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
<b>CONTROL</b>										
EI	Enable interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupts	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4
HLT	Halt	0	1	1	1	0	1	1	0	5
<b>NEW 8085 INSTRUCTIONS</b>										
RIM	Read interrupt mask	0	0	1	0	0	0	0	0	4
SIM	Set interrupt mask	0	0	1	1	0	0	0	0	4

**Notes**

1. All mnemonics copyright © Intel Corporation 1977.
2. DDD or SSS: B=000, C=001, D=010, E=011, H=100, L=101, Memory=110, A=111.
3. Two possible cycle times. (6/12) indicates instruction cycles dependent on condition flags.

**Table 7-7. Summary of 8085A Processor Instructions**

**SPECIFICATIONS**

**Central Processor**

**CPU — 8085A**

**Instruction Cycle — 1.3 μs**

**Tcy — 330 ns**

**Memory**

**ROM — 2K bytes (expandable to 4K bytes) 8355/8755A**

**RAM — 256 bytes (expandable to 512 bytes) 8155**

**Addressing**

**ROM — 0000-07FF (expandable to 0FFF with an additional 8355/8755A)**

**RAM — 2000-20FF (2800-28FF available with an additional 8155)**

**Note**

The wire-wrap area of the SDK-85 PC board may be used for additional custom memory expansion up to the 64K-byte addressing limit of the 8085A.





FORTRAN-80

FORTRAN-80 LANGUAGE FEATURES

## **FORTRAN-80 8080/8085 ANS FORTRAN 77 INTELLEC® RESIDENT COMPILER**

**Meets and exceeds ANS FORTRAN 77  
Subset Language Specification**

**Supports Intel Floating Point Standard  
with either the floating point support  
library or the iSBC-310 High Speed  
Mathematics Board**

**Resident operation on Inteltec®  
Microcomputer Development System  
and Inteltec® Series II Microcomputer  
Development System**

**Supports full symbolic debugging with  
ICE-80™ and ICE-85™**

**Produces relocatable and linkable object  
code compatible with resident PL/M-80  
and 8080/8085 Macro Assembler**

**Full FORTRAN 77 language I/O support or  
optional RMX-80 run-time library**

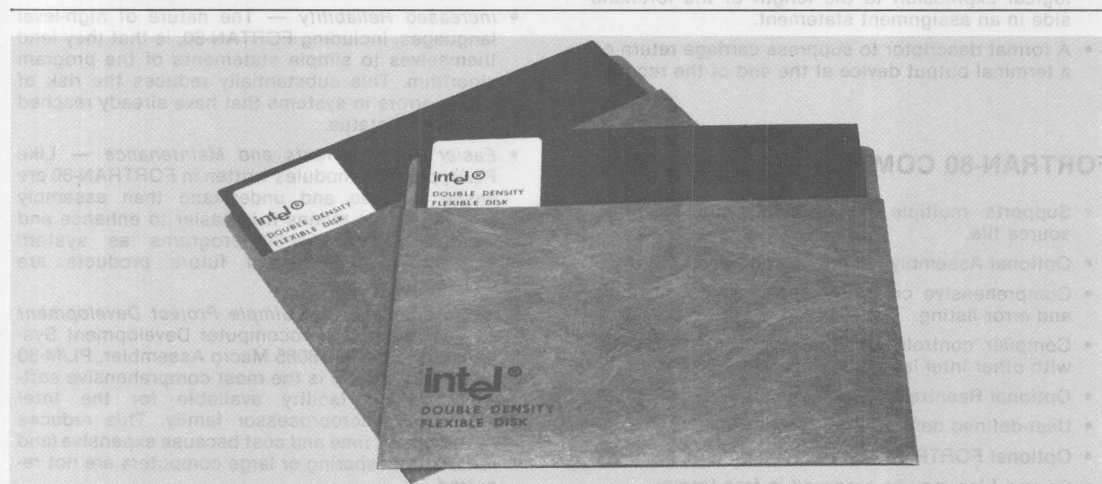
**Well defined I/O interface for configu-  
ration with user-supplied drivers**

**Sophisticated code optimization insures  
efficient program implementation**

FORTRAN-80 is a computer industry-standard, high-level programming language and compiler that translates FORTRAN statements into relocatable object modules. When the object modules are linked together and located into absolute program modules, they are suitable for execution on Intel® 8080/8085 Microprocessors, iSBC-80 OEM Computer Systems, and Inteltec® Microcomputer Development Systems. FORTRAN-80 meets and exceeds the ANS FORTRAN 77 Language Subset Specification<sup>1</sup>. The compiler operates on the Inteltec Microcomputer Development System under the ISIS-II Disk Operating Systems and produces efficient relocatable object modules that are compatible for linkage with PL/M-80 and 8080/8085 Macro Assembler modules.

The ANS FORTRAN 77 language specification offers many powerful extensions to the FORTRAN language that are especially well suited to Intel® 8080/8085 Microprocessor software development. Because FORTRAN-80 conforms to the ANS FORTRAN 77 standard, the user is assured of compatibility with existing FORTRAN software that meets the standard as well as a guarantee of upward compatibility to other computer systems supporting an ANS FORTRAN 77 Compiler.

<sup>1</sup>ANSI X3J3/90





## FORTRAN-80

- Structured Programming is supported with the IF ... THEN ... ELSE IF ... ELSE ... END IF constructs.
- CHARACTER data type permits alphanumeric data to be handled as strings rather than characters stored in array elements.
- Full I/O capabilities include:
  - Sequential and Direct Access files
  - Error handling facilities
  - Formatted, Free-formatted, and Unformatted data representation
  - Internal (in-memory) file units provide capability to format and reformat data in internal memory buffers
  - List Directed Formatting
- Supports arrays of up to seven dimensions.
- Supports logical operators
  - .EQV. — Logical equivalence
  - .NEQV. — Logical nonequivalence

Major extensions to FORTRAN 77 in Intel FORTRAN-80 include:

- Direct 8080/8085 port I/O supported by intrinsic subroutines.
- Binary and Hexadecimal integer constants.
- Well defined interface to FORTRAN-80 I/O statements (READ, OPEN, etc.), allowing easy use of user-supplied I/O drivers.
- User-defined INTEGER storage lengths of 1, 2 or 4 bytes.
- User-defined LOGICAL storage lengths of 1, 2 or 4 bytes.
- REAL STORAGE lengths of 4 bytes.
- Bitwise Boolean operations using logical operators on integer values.
- Hollerith data constants.
- Implicit extension of the length of an integer or logical expression to the length of the left-hand side in an assignment statement.
- A format descriptor to suppress carriage return on a terminal output device at the end of the record.

## FORTRAN-80 COMPILER FEATURES

- Supports multiple compilation units in single source file.
- Optional Assembly Language code listing.
- Comprehensive cross-reference, symbol attribute and error listing.
- Compiler controls and directives are compatible with other Intel language translators.
- Optional Reentrancy.
- User-defined default storage lengths.
- Optional FORTRAN 66 Do Loop semantics.
- Source files may be prepared in free format.

floating point support, allowing either to be chosen at time of linking.

## FORTRAN-80 BENEFITS

FORTRAN-80 provides a means of developing application software for Intel® MCS-80/85 products in a familiar, widely accepted, and computer industry-standardized programming language. FORTRAN-80 will greatly enhance the user's ability to provide cost-effective solutions to software development for Intel microprocessors as illustrated by the following:

- *Completely Complementary to Existing Intel Software Design Tools* — Object modules are linkable with new or existing Assembly Language and PL/M Modules.
- *Incremental Runtime Library Support* — Runtime overhead is limited only to facilities required by the program.
- *Low Learning Effort* — FORTRAN-80, like PL/M, is easy to learn and use. Existing FORTRAN software can be ported to FORTRAN-80, and programs developed in FORTRAN-80 can be run on any other computer with ANS FORTRAN 77.
- *Earlier Project Completion* — Critical projects are completed earlier than otherwise possible because FORTRAN-80 will substantially increase programmer productivity, and is complementary to PL/M Modules by providing comprehensive arithmetic, I/O formatting, and data management support in the language.
- *Lower Development Cost* — Increases in programmer productivity translates into lower software development costs because less programming resources are required for a given function.
- *Increased Reliability* — The nature of high-level languages, including FORTRAN-80, is that they lend themselves to simple statements of the program algorithm. This substantially reduces the risk of costly errors in systems that have already reached production status.
- *Easier Enhancements and Maintenance* — Like PL/M, program modules written in FORTRAN-80 are easier to read and understand than assembly language. This means it is easier to enhance and maintain FORTRAN-80 programs as system capabilities expand and future products are developed.
- *Comprehensive, Yet Simple Project Development* — The Intellec Microcomputer Development System, with the 8080/8085 Macro Assembler, PL/M-80 and FORTRAN-80 is the most comprehensive software design facility available for the Intel MCS-80/85 Microprocessor family. This reduces development time and cost because expensive (and remote) timesharing or large computers are not required.

\*\*\* THIS PROGRAM IS AN EXAMPLE OF ISIS-II FORTRAN-80 THAT  
 \*\* CONVERTS TEMPERATURE BETWEEN CELSIUS AND FARENHEIT

# PROGRAM CONVRT

CHARACTER\*1 CHOICE, SCALE

PRINT 100

\*\* ENTER CONVERSION SCALE (C OR F)

PRINT 200

READ (5,300) SCALE

IF (SCALE .EQ. 'C')

THEN

PRINT 400

\*\* ENTER THE NUMBER OF DEGREES FARENHEIT

READ (5,\*) DEGF

DEGC = 5./9.\*(DEGF-32)

\*\* PRINT THE ANSWER

WRITE (6,500) DEGF,DEGC

\*\* RUN AGAIN?

PRINT 600

READ (5,300) CHOICE

IF (CHOICE .EQ. 'Y')

THEN

GOTO 10

ELSE IF (CHOICE .EQ. 'N')

THEN

CALL EXIT

ELSE

GOTO 20

END IF

ELSE IF (SCALE .EQ. 'F')

THEN

\*\* CONVERT FROM FARENHEIT TO CELSIUS

PRINT 700

READ (5,\*) DEGC

DEGF = 9./5.\*DEGC+32.

\*\* PRINT THE ANSWER

WRITE (6,800) DEGC,DEGF

GOTO 20

ELSE

\*\* NOT A VALID ENTRY FOR THE SCALE

WRITE (6,900) SCALE

GOTO 10

END IF

100 FORMAT(' TEMPERATURE CONVERSION PROGRAM',//,

+' TYPE C FOR FARENHEIT TO CELSIUS OR',/,

+' TYPE F FOR CELSIUS TO FARENHEIT',//)

200 FORMAT(/, ' CONVERSION? ', \$)

300 FORMAT(A1)

400 FORMAT(/, ' ENTER DEGREES FARENHEIT: ', \$)

500 FORMAT(/, F7.2, ' DEGREES FARENHEIT = ', F7.2, ' DEGREES CELSIUS.')

600 FORMAT(/, ' AGAIN (Y OR N)? ', \$)

700 FORMAT(/, ' ENTER DEGREES CELSIUS: ', \$)

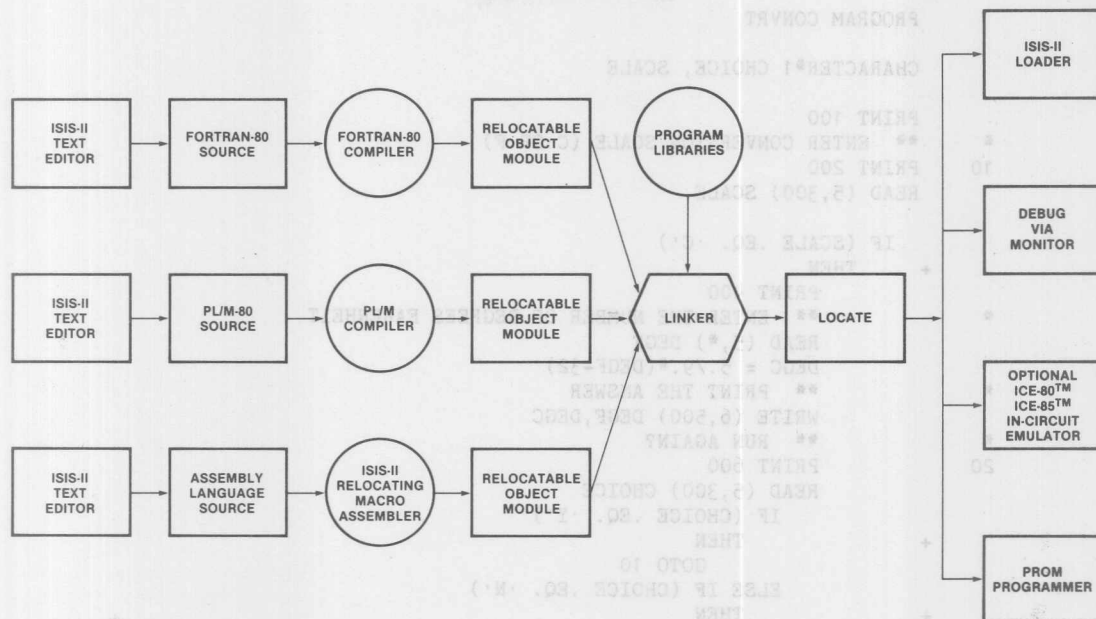
800 FORMAT(/, F7.2, ' DEGREES CELSIUS = ', F7.2, ' DEGREES FARENHEIT', /)

900 FORMAT(/, 1H ,A1, ' NOT A VALID CHOICE - TRY AGAIN!', /)

END

# FORTRAN-80

The FORTRAN-80 Compiler is an efficient, multiphase compiler that accepts source programs, translates them into relocatable object code, and produces requested listings. After compilation, the object program may be linked to other modules, located to a specific area of memory, then executed. The diagram shown below illustrates a program development cycle where the program consists of modules created by FORTRAN-80, PL/M-80 and the 8080/8085 Macro Assembler.



## SPECIFICATIONS

### OPERATING ENVIRONMENT

#### Required Hardware:

- Intellec® Microcomputer Development System
  - MDS-800, MDS-888
  - Series II Model 220, Model 230

64K bytes of RAM memory

Dual diskette drives

- Single or Double Density

System console

- CRT or hardcopy interactive device

#### Optional Hardware:

- Line Printer
- ICE-80™, ICE-85™

#### Required Software:

- ISIS-II Diskette Operating System
  - Single or Double Density

#### Optional Software:

- ISBC-801 FORTRAN-80 Run-Time Software Package for RMX-80

### DOCUMENTATION PACKAGE

FORTRAN-80 Programming Manual (9800481)

ISIS-II FORTRAN-80 Compiler Operator's Manual (9800480)

FORTRAN-80 Programming Reference Card (9800547)

### SHIPPING MEDIA

Flexible Diskettes

- Single and Double Density

## ORDERING INFORMATION

PRODUCT CODE	DESCRIPTION
MDS-301	FORTRAN-80 Compiler for Intellec Microcomputer Development Systems



## **BASIC-80 EXTENDED ANS 1978 BASIC INTELLEC® RESIDENT INTERPRETER**

**Meets ANS 1978 standard for minimal BASIC and adds many powerful extensions**

**Operates under the ISIS-II operating system on Intellec and Intellec® Series-II Microcomputer Development Systems**

**Full sequential and random disk file I/O with ISIS-II**

**Applications range from prototyping microcomputer software to inexpensive engineering and management problem solving on the Intellec® systems**

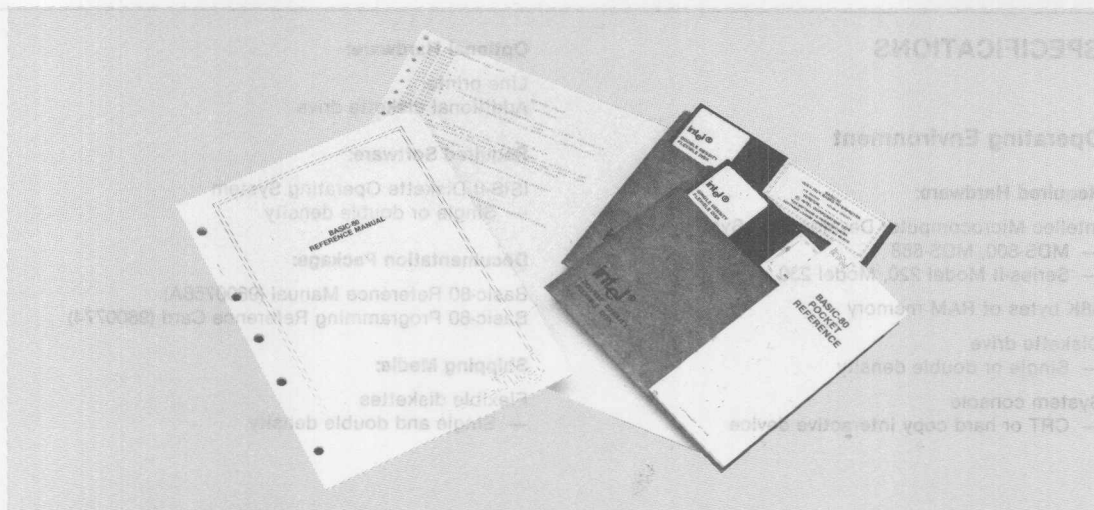
BASIC is an industry standard, high-level programming language which is designed to be easily learned and used by novices and experienced programmers alike. The interpreter provides an interactive environment which allows fast and easy program development, testing, and debugging. BASIC is widely used for problem solving in engineering and management; extensive software exists for business applications such as order entry, accounts receivable, accounts payable, and inventory control, and engineering applications such as numeric and statistical analysis.

Intel's BASIC-80 meets the standards of ANS 1978 BASIC and extends them to take advantage of the software development capabilities of the Intellec Microcomputer Development Systems. The matching of these resources with the ease of programming in BASIC-80 provides a very effective tool for both microprocessor systems development and inexpensive applications programming and problem solving on the Intellec systems.

**Supports the Intel floating point standard and provides integer and string data types**

**Can call user subroutines written in FORTRAN-80, PL/M-80, and 8080/85 macro assembler that are resident in the Intellec® memory**

**Easily learned language and interactive environment combine to provide a flexible and powerful facility for developing programs to run on the Intellec® Microcomputer Development Systems**



## BASIC-80

- String and numeric constants, variables, and arrays.
- FOR...TO...STEP...NEXT statements for loop execution.
- IF...THEN statements for conditional execution.
- ON...GOTO statements for computed branching.
- GOSUB/RETURN subroutine calls and returns.
- Built in scientific functions:

ABS	RND	TAN
EXP	SGN	COS
INT	SQR	SIN
LOG	ATN	

- User defined single statement functions.

Major extensions to ANS 78 BASIC which BASIC-80 provides include:

- Support for the Intel single and double precision floating point standard.
- Disk file I/O, supporting both random access and sequential access files.
- Direct read and write to CPU I/O ports through the INP and OUT functions.
- Direct memory read and write through the PEEK and POKE functions.
- Calls to user-supplied external subroutines, which may have been written in FORTRAN-80, PL/M-80, or 8080/8085 Assembly Language and have been located at absolute memory locations using the ISIS-II facilities.
- User directed error trapping and handling functions.
- Program execution trace command.

- Matrices with up to 110 dimensions.
- Extensive string manipulation functions.
- Boolean operators.
- Type conversion functions—integer, floating point, and character.

## BENEFITS OF BASIC-80

- Added Value to the Intellec Systems—with BASIC-80 the Intellec Microcomputer Development Systems can be effectively used in many engineering and management applications.
- Inexpensive and Accessible Computational Facility—the ease of use and flexibility inherent in BASIC-80 and its interpretive environment fit well with the "at hand" computational resources of the Intellec systems. The combination is a particularly useful tool for obtaining fast and accurate results.
- Easy to Learn—the language is designed to be easily understood and learned. Results are obtained faster and people who may benefit from using the system can do so easily.
- Aid in Microcomputer Software Design—microcomputer software can be prototyped in BASIC-80 to inexpensively develop and test program logic.
- Complemented by Existing Software—subroutines written in PL/M-80, FORTRAN-80, and ASM 8080/85 can be called from BASIC-80 programs.
- Easy to Enhance and Maintain—BASIC-80, being straightforward and easily understood, provides for programs that are easy to maintain and modify in the future.

## SPECIFICATIONS

### Operating Environment

#### Required Hardware:

Intellec Microcomputer Development System  
— MDS-800, MDS-888  
— Series-II Model 220, Model 230

48K bytes of RAM memory

Diskette drive

— Single or double density

System console

— CRT or hard copy interactive device

#### Optional Hardware:

Line printer

Additional diskette drive

#### Required Software:

ISIS-II Diskette Operating System

— Single or double density

#### Documentation Package:

Basic-80 Reference Manual (9800758A)

Basic-80 Programming Reference Card (9800774)

#### Shipping Media:

Flexible diskettes

— Single and double density



## EXAMPLE BASIC-80 PROGRAM

```

list
10 PRINT "THIS PROGRAM CALCULATES THE MEAN AND STANDARD"
20 PRINT " DEVIATION OF INPUT DATA"
30 S=0:V=0
40 INPUT "NUMBER OF VALUES";N
50 FOR I=1 TO N
60 INPUT A(I)
70 S=S+A(I)
80 NEXT
90 S=S/N
100 REM CALCULATION OF VARIANCE
110 FOR I=1 TO N
120 V=V+(A(I)-S)**2/N
130 NEXT
140 SD=SQR(V)
150 PRINT "MEAN=";S
160 PRINT "STANDARD DEVIATION IS=";SD
Ok

```

```

run
THIS PROGRAM CALCULATES THE MEAN AND STANDARD
DEVIATION OF INPUT DATA
NUMBER OF VALUES? 6
? 34.7
? 32.9
? 38.2
? 35
? 37.6
? 40.9
MEAN= 36.55
STANDARD DEVIATION IS= 2.642442
Ok

```

## ORDERING INFORMATION

Product Code	Description
MDS-320	ISIS-II BASIC-80 Disk-Based Interpreter



## iCIS-COBOL™ SOFTWARE PACKAGE

**Meets and exceeds minimum ANSI Level 1 standard for COBOL (X3.23-1974).**

**Runs under ISIS-II on Inteltec or Inteltec® Series II Microcomputer Development Systems.**

**Compiler compiles COBOL source programs into an intermediate code which is optimized for speed and memory space.**

**Includes execution run-time interpreter and an interactive debugger.**

**Powerful extensions for interactive programming.**

**Can Link/Call routines written in PL/M-80, FORTRAN-80 and 8080/8085 Assembly Language.**

**FORMS utility program allows the user to design and test CRT screen format input by generating COBOL source code for the data descriptions defining that CRT screen format.**

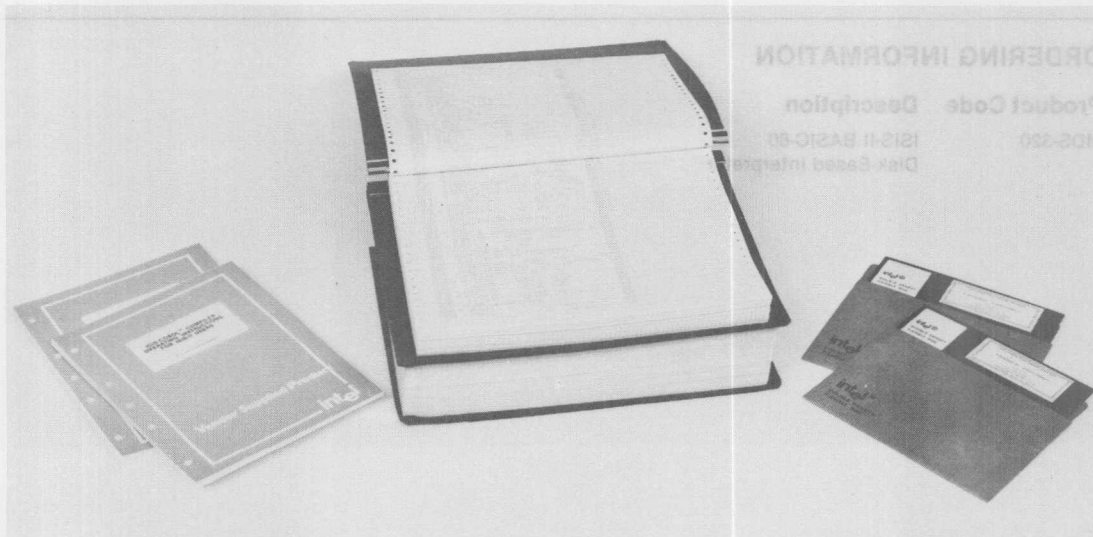
**Compile-time option available to flag any non-ANSI standard features for portability.**

**Tested using U.S. Dept. of Navy COBOL validation system.**

iCIS-COBOL, an acronym for Intel's Compact Interactive Standard COBOL, is a package designed to provide a powerful interactive business language to users of Intel's Inteltec and Inteltec Series II Microcomputer Development Systems. iCIS-COBOL contains the most relevant parts of the ANSI 74 standard plus extra extensions to make this product especially useful to Inteltec users. The compiler provides a feature to optionally disallow the iCIS-COBOL extensions and rigidly enforce the ANSI 74 specification. This will prove beneficial to users who may need to port COBOL programs from the Inteltec system to any other ANSI Level 1 COBOL compiler.

iCIS-COBOL Compiler generates object code for a COBOL "virtual machine." This code is designed for optimum representation of COBOL verbs and data types. The code generated is interpreted by a Run-Time System. This consists of an interpreter which emulates the COBOL virtual machine and interfaces to the ISIS-II operating system and the CRT.

After an application program has been tested and is ready for production use, it is possible to link it permanently to the Run-Time System to form a free-standing ISIS-II loadable program.



## LANGUAGE FEATURES

COBOL consists of twelve different modules implemented either to Level 1 or Level 2 as defined in the ANSI specification X3.23. iCIS-COBOL includes the following modules implemented to Level 1:

- Nucleus
- Table Handling
- Sequential I/O
- Relative I/O
- Indexed I/O
- Library
- Interprogram Communication

Extensions to ANSI Specification:

- **Advanced screen formatting and data entry facilities.** These include protected and unprotected data, cursor manipulation, and numeric vet.
- **Run time input of filenames.** The actual value of the external filename may be moved to a file identifier location prior to OPENing the file, avoiding the need for an external linking mechanism.
- **Line sequential files.** Variable length records separated by carriage return/line feed saves space on disk and allows iCIS-COBOL programs to process files output by a text editor.
- **Hexadecimal literals.** These may be used to define control characters to output to special peripheral devices.
- **Rapid development facilities.** During development, compiled programs may be loaded directly by the Run-Time System "fast load" facility, thus avoiding the time otherwise spent in linking.
- **Interactive debugging.** Interactive debugging permits the setting of breakpoints, examination and modification of store, etc., at run time. Each COBOL statement is identified by a four-digit hexadecimal number.
- **Lower case.** This is permitted in COBOL words and comments, thus helping to produce easy to read documentation in the program.

## INTERACTIVE CRT HANDLING

Intel has taken COBOL — traditionally a batch processing language — and extended it to become interactive. iCIS-COBOL offers many facilities for automatically formatting a CRT screen and facilitating input keying.

The user can format the screen of any system console (CRT) into protected and unprotected fields by using standard COBOL statements. The screen layout may be defined in the DATA DIVISION. An ACCEPT statement nominates a record description which permits input to the character positions corresponding to variables identified by data-names. These may be separated by FILLERS to position them on the screen. Conversely, a DISPLAY outputs only from non-FILLER fields in the record description which it nominates. The programmer can easily build up complex conversations for data entry and transaction processing.

When data is being keyed in, the operator has full cursor manipulation facilities, each variable acting as a tab stop. Non-numeric digits may not be keyed into fields defined as PIC 9. Finally, when the operator has checked that the data is correct, the RETURN key is pressed and processing continues.

### SCREEN LAYOUT AND FORMAT FACILITIES

Screen as a record description

FILLER

REDEFINES

AT line:column

Character highlighting

Clear screen

Numeric vet for PIC fields

### CURSOR CONTROL FACILITIES

HOME to the start of the first data field on the screen

→ Forward space

← Backward space

↓ Forward field

↑ Backward field

C/R Release the screen of data

L/F Left Fill numeric field

(The actual keys used vary according to CRT keyboard)

## FORMS UTILITY

A majority (up to 80%) of debugging time can be spent in designing, coding and testing the screen form input/output of a COBOL program. The FORMS utility included in the iCIS-COBOL package significantly reduces this debugging time.

Using the FORMS program, the user may:

- Store an image copy on disk of the form he has defined for subsequent use.
- Generate iCIS-COBOL source code for the data descriptions required to define the form just created. This may then be included in an iCIS-COBOL program using COPY.
- Choose to generate a checkout program which allows duplication of the many machine conversations which would take place during a run of the application which is being designed.

## COMPILE TIME DIRECTIVES

### • ANS

If specified, the Compiler will accept only those iCIS-COBOL language statements that conform to the ANS 74 standard.

### • RESEQ

If specified, the Compiler generates COBOL sequence numbers, renumbering each line in increments of 10.

### • NOINT

No intermediate code file is output. The Compiler is, in effect, used for syntax checking only.

### • NOLIST

No list file is produced; used for fast compilation of "clean" programs.

# ICIS-COBOL

No form feed or page headings are to be output by the Compiler in the list file.

## • ERRLIST

The listing is limited to those COBOL lines containing syntax errors together with the associated error message(s).

## • INT (external-file-name)

Specifies the file to which the intermediate code is to be directed.

## • LIST (external-file-name)

Specifies the file to which the listing is to be directed.

## • FORM (integer)

Specifies the number of COBOL lines per page of listing.

## • NOECHO

Error lines are echoed on the console unless this directive is specified.

## BENEFITS

- Brings COBOL to Intelc Microcomputer Development Systems.

— COBOL is the industry standard high-level language for business-oriented applications.

- More business and application programs are written in COBOL than any other language.

- Meets and exceeds ANS Level 1 COBOL standard.

— Assures portability to and from all computers supporting ANS Level 1 COBOL.

— Extensive testing and validation using U.S. NAVY COBOL VALIDATION SYSTEM assures functionality for all Level 1 features.

- iCIS-COBOL software package provides an easy to use, efficient and friendly environment for COBOL program development.

— CONFIGURATOR program allows the user to reconfigure the software for any non-standard, non-Intel CRT.

— Interactive debugger provides features aimed at a CRT based system (rather than batch-oriented).

— FORMS utility program reduces total program development time by 30%.

— All iCIS-COBOL utilities make use of CRT cursor control.

- Adds value to an Intelc development system.

— COBOL applications programs developed using iCIS-COBOL software package will increase utilization of Intelc development systems.

The source program is created on diskette with the iCIS-ii editor.

COBOL PROG.SRC....

... Loads the single-pass compiler to convert the source program into an interpreted object form known as intermediate code. The user may specify the file on which the listing will appear. If this is a disk file, then it may be edited to correct errors and used as input for the next run of the compiler.

RUN PROG.INT....

... Loads the Run-Time System, which in turn loads the program. During program development, the two commands COBOL and RUN are sufficient to produce the program. To aid debugging, the iCIS-COBOL interactive debugging facility is available. This allows the user to set breakpoints, examine and modify locations, and then continue execution. Note that before running a program that uses ACCEPT or DISPLAY, the system requires configuration for a non-Intel CRT.

RUN = PROG.INT....

... Once your program is fully tested, it may be permanently linked to the Run-Time System by using the "L" option. This produces a file named "SAVE" that may be directly executed.

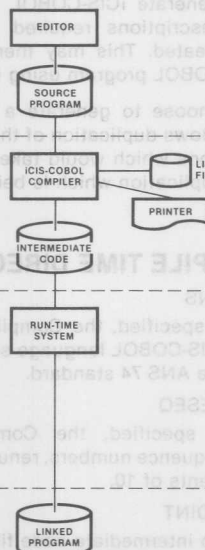


Figure 7-12. Program Development Cycle

```

**ICIS-COBOL V 1.0
PAGE: 0001

000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. STUCK-PILE-GET-UP.
000030 AUTHOR. RALPH PETER.
000040 ENVIRONMENT DIVISION.
000050 CONFIGURATION SECTION.
000060 SOURCE-PROGRAM-TEMP.
000070 OBJECT-COUPLES.
000080 INTERPRETER-SECTION.
000090 FILE-CONTROL.
000100 SELECT STUCK-PILE ASSIGN TO STUCK-IT.
000110 GENERAL-EDITION INSTRUCTIONS.
000120 ACCEPTED INSTRUCTIONS.
000130 RECALL-AND-AT-STUCK-CODE.
000140 DATA DIVISION.
000150 FILE SECTION.
000160 FD STUCK-PILE RECORDS 32.
000170 STUCK-PILE.
000180 SD STUCK-CODES PIC A(10).
000190 SD STUCK-PILE PIC A(10).
000200 SD STUCK-PILE PIC A(10).
000210 STUCK-PILE.
000220 SD STUCK-PILE PIC A(10).
000230 SD STUCK-PILE PIC A(10).
000240 SD STUCK-PILE PIC A(10).
000250 SD STUCK-PILE PIC A(10).
000260 SD STUCK-PILE PIC A(10).
000270 SD STUCK-PILE PIC A(10).
000280 SD STUCK-PILE PIC A(10).
000290 SD STUCK-PILE PIC A(10).
000300 SD STUCK-PILE PIC A(10).
000310 SD STUCK-PILE PIC A(10).
000320 SD STUCK-PILE PIC A(10).
000330 SD STUCK-PILE PIC A(10).
000340 SD STUCK-PILE PIC A(10).
000350 SD STUCK-PILE PIC A(10).
000360 SD STUCK-PILE PIC A(10).
000370 SD STUCK-PILE PIC A(10).
000380 SD STUCK-PILE PIC A(10).
000390 SD STUCK-PILE PIC A(10).
000400 SD STUCK-PILE PIC A(10).
000410 SD STUCK-PILE PIC A(10).
000420 SD STUCK-PILE PIC A(10).
000430 SD STUCK-PILE PIC A(10).
000440 SD STUCK-PILE PIC A(10).
000450 SD STUCK-PILE PIC A(10).
000460 SD STUCK-PILE PIC A(10).
000470 SD STUCK-PILE PIC A(10).
000480 SD STUCK-PILE PIC A(10).
000490 SD STUCK-PILE PIC A(10).
000500 SD STUCK-PILE PIC A(10).
000510 SD STUCK-PILE PIC A(10).
000520 SD STUCK-PILE PIC A(10).
000530 SD STUCK-PILE PIC A(10).
000540 SD STUCK-PILE PIC A(10).
000550 SD STUCK-PILE PIC A(10).
000560 SD STUCK-PILE PIC A(10).
000570 SD STUCK-PILE PIC A(10).
000580 SD STUCK-PILE PIC A(10).
000590 SD STUCK-PILE PIC A(10).
000600 SD STUCK-PILE PIC A(10).
000610 SD STUCK-PILE PIC A(10).
000620 SD STUCK-PILE PIC A(10).
000630 SD STUCK-PILE PIC A(10).
000640 SD STUCK-PILE PIC A(10).
000650 SD STUCK-PILE PIC A(10).
000660 SD STUCK-PILE PIC A(10).
000670 SD STUCK-PILE PIC A(10).
000680 SD STUCK-PILE PIC A(10).
000690 SD STUCK-PILE PIC A(10).
000700 SD STUCK-PILE PIC A(10).
000710 SD STUCK-PILE PIC A(10).
000720 SD STUCK-PILE PIC A(10).
000730 SD STUCK-PILE PIC A(10).
000740 SD STUCK-PILE PIC A(10).
000750 SD STUCK-PILE PIC A(10).
000760 SD STUCK-PILE PIC A(10).
000770 SD STUCK-PILE PIC A(10).
000780 SD STUCK-PILE PIC A(10).
000790 SD STUCK-PILE PIC A(10).
000800 SD STUCK-PILE PIC A(10).
000810 SD STUCK-PILE PIC A(10).
000820 SD STUCK-PILE PIC A(10).
000830 SD STUCK-PILE PIC A(10).
000840 SD STUCK-PILE PIC A(10).
000850 SD STUCK-PILE PIC A(10).
000860 SD STUCK-PILE PIC A(10).
000870 SD STUCK-PILE PIC A(10).
000880 SD STUCK-PILE PIC A(10).
000890 SD STUCK-PILE PIC A(10).
000900 SD STUCK-PILE PIC A(10).
000910 SD STUCK-PILE PIC A(10).
000920 SD STUCK-PILE PIC A(10).
000930 SD STUCK-PILE PIC A(10).
000940 SD STUCK-PILE PIC A(10).
000950 SD STUCK-PILE PIC A(10).
000960 SD STUCK-PILE PIC A(10).
000970 SD STUCK-PILE PIC A(10).
000980 SD STUCK-PILE PIC A(10).
000990 SD STUCK-PILE PIC A(10).
001000 SD STUCK-PILE PIC A(10).
  
```

Figure 7-13. Sample Program Listing Showing Source Format

## SPECIFICATIONS

### Operating Environment

#### Required Hardware:

Intellec Microcomputer Development System

— Model 800

— Series II Model 220, Model 230

48KB of Memory

Dual Diskette Drives

— Single or Double Density

System Console

— Intel or non-Intel CRT

#### Recommended Hardware:

64KB of Memory

Double Density Dual Diskette Drives

#### Optional Hardware:

Line Printer

#### Required Software:

ISIS-II Diskette Operating System

— Single or Double Density

#### Optional Software:

ISIS-II CREDIT (CRT-Based Text Editor)

### Documentation Package

iCIS-COBOL Language Reference Manual (9800927-01)

iCIS-COBOL Compiler Operating Instructions for ISIS-II Users (9800928-01)

iCIS-COBOL Pocket Reference (9800929-01)

### Shipping Media

Flexible Diskettes

— Single and Double Density

---

## ORDERING INFORMATION:

Product Code	Description
MDS-380	iCIS-COBOL Software Package





# Appendix Applications of MCS-85™

A1-1  
A1-2  
A1-3

A1-4

A1-5  
A1-6  
A1-7  
A1-8  
A1-9  
A1-10  
A1-11  
A1-12  
A1-13  
A1-14  
A1-15  
A1-16  
A1-17  
A1-18  
A1-19  
A1-20  
A1-21  
A1-22  
A1-23  
A1-24

A1-25  
A1-26  
A1-27  
A1-28  
A1-29  
A1-30  
A1-31  
A1-32  
A1-33  
A1-34  
A1-35  
A1-36  
A1-37  
A1-38  
A1-39  
A1-40  
A1-41  
A1-42  
A1-43  
A1-44  
A1-45  
A1-46

A1-47  
A1-48  
A1-49  
A1-50  
A1-51  
A1-52  
A1-53  
A1-54  
A1-55  
A1-56  
A1-57  
A1-58  
A1-59  
A1-60  
A1-61  
A1-62  
A1-63  
A1-64  
A1-65  
A1-66  
A1-67  
A1-68  
A1-69  
A1-70  
A1-71  
A1-72  
A1-73  
A1-74  
A1-75  
A1-76  
A1-77  
A1-78  
A1-79  
A1-80  
A1-81  
A1-82  
A1-83  
A1-84  
A1-85  
A1-86  
A1-87  
A1-88  
A1-89  
A1-90  
A1-91  
A1-92  
A1-93  
A1-94  
A1-95  
A1-96  
A1-97  
A1-98  
A1-99  
A1-100

# Appendix

## Applications of MCS-85™

### Contents

---

#### SECTION 1 INTRODUCTION TO MCS-85™ APPLICATIONS

MCS-85™ System .....	A1-1
Sample Applications .....	A1-2
Baud Rate Generator .....	A1-2
Serial Communications .....	A1-2
Small System .....	A1-6
Block Move, Block Search .....	A1-6
RST 7 .....	A1-7

#### SECTION 2 DETAILED APPLICATION EXAMPLES

Memory Addressing .....	A1-8
ROM, EPROM .....	A1-8
Static Memories .....	A1-11
Dynamic RAM Interface .....	A1-11
<b>System Timings</b> .....	A1-16
8085A CLK-IN vs. CLK-OUT vs. Control Timings .....	A1-16
3.125 vs. 5 MHz Considerations .....	A1-18
Memory Device Compatibility .....	A1-20
Peripheral Compatibility—3.125 and 5 MHz .....	A1-23
Bus - Loading Considerations - Decoupling .....	A1-25

#### APPLICATION EXAMPLE 1

<b>Minimum System Application Example as a Temperature Sensor</b> .....	A1-26
Overview .....	A1-26
Detailed Hardware .....	A1-26
Software Block Move, Search Illustrations .....	A1-29

#### APPLICATION EXAMPLE 2

<b>CRT Interface</b> .....	A1-32
Hardware Interface .....	A1-32
Software Package .....	A1-32
Output Routine .....	A1-34
Input Routine .....	A1-35
Timing Analysis .....	A1-35
Baud Rate Identification Routine .....	A1-36

#### APPLICATION EXAMPLE 3

<b>Cassette Recorder Interface</b> .....	A1-38
Hardware Design .....	A1-38
Software .....	A1-38
Output Routine .....	A1-40
Input Routine .....	A1-40
<b>Additional Comments</b> .....	A1-42
<b>Temperature Sensor Code</b> .....	A1-44
<b>CRT and Cassette Code</b> .....	A1-48

# APPENDIX 1 APPLICATIONS OF MCS-85™

## SECTION 1 INTRODUCTION TO MCS-85™ APPLICATIONS

When the first microprocessor was introduced about five years ago, it was largely ignored by the electronics industry. However, since that inauspicious beginning, this new device has become the hottest topic in current technology. As more and more product designers become familiar with the capabilities of microcomputers, the number of new applications increases geometrically. In most of these applications, the new technology has been used to replace designs which were formerly implemented with TTL logic and under-utilized minicomputers. However, an increasing number of products are surfacing which would have been impractical prior to the microcomputer era.

Microcomputers are being applied to a wide range of data communications tasks. The field of telephone equipment is being invaded by systems which control and monitor calls. Point of sale terminals are increasing daily with the addition of interface to coin changers, electronic scales and remote computers. Small stand-alone computers are relying heavily upon microcomputers in teleprocessing, time-sharing, data base management and similar interactive applications. An increasing number of microcomputer-based data terminals are providing local interactive intelligence with programmable character sets, vector generation and the pre-processing of data.

Instrumentation is widely utilizing the microprocessor for a variety of control and arithmetic processing functions. Microcomputers are controlling laboratory equipment such as oscilloscopes, DVM's, network analyzers and frequency synthesizers. Medical electronics are crediting microcomputers with tasks such as patient monitoring, blood analysis and X-ray scanning. Travel is becoming microcomputerized by automotive control, air and ocean navigation equipment and rapid transit systems.

### MCS-85™ SYSTEM

Many possible microcomputer applications have been overlooked because of the design tasks required to build the microcomputer. These tasks include the system clock, read/write memory, I/O ports, serial communications interface and bus control logic. The MCS-85 system will enable the design engineer to concentrate on the application of the microcomputer, rather than on the implementation details.

The MCS-85 is yet another family of components which has the potential to provide a solution to the three problems which will always plague designers: cost, size and power. The reduced component count of an MCS-85 microcomputer, coupled with the increased integration of functions reduces both cost and size while increasing power.

BAUD RATE	COUNT (DECIMAL)
300	10240
600	5120
1200	2560
2400	1280
4800	640
9600	320

FIGURE 1. BAUD RATES

## Sample Applications

Calculating Oscilloscope  
Blood Analyzer  
Programmable Video Game  
Process Control System  
Line Printer

Intelligent Terminal  
N.C. Machine  
Digital Multimeter  
Graphic Terminal  
Automotive Control

Navigation Equipment  
Vending Machine  
Spectrum Analyzer  
Front End Processor  
Credit Verifier

Disk Controller  
Patient Monitor  
Network Analyzer  
Frequency Synthesizer

APPLICATION	PERIPHERAL DEVICES ENCOUNTERED	MCS-85™ COMPONENTS	
Intelligent Terminals	Cathode Ray Tube Display Printing Units Synchronous and Asynchronous data lines Cassette Tape Unit Keyboards	8275	8085A
		8155	8355
Gaming Machines	Keyboards, pushbuttons and switches Various display devices Coin acceptors Coin dispensers	8251	
		8279	
Cash Registers	Keyboard or Input Switch Array Change Dispenser Digital Display Ticket Printer Magnetic Card reader Communication interface	8279	8085A
		8155	8355
Accounting and Billing Machines	Keyboard Printer Unit Cassette or other magnetic tape unit "Floppy" disks	8273	
		8279	8085A
Telephone Switching Control	Telephone Line Scanner Analog Switching Network Dial Registers Class of Service Parcel	8155	8355
		8155	
Numerically Controlled Machines	Magnetic or Paper Tape Reader Stepper Motors Optical Shaft Encoders	8155	8085A
			8355
Process Control	Analog-to-Digital Converters Digital-to-Analog Converters Control Switches Displays	8155	8085A
		8279	8355

## Baud Rate Generator

Shown in Figure 2 is a minimum system configuration with the 8156 timer output connected to an 8085 interrupt input.

This configuration allows convenient use of the timer as a baud rate generator. A 6.144 MHz crystal is used as the frequency control element of the 8085A, providing integral divisors for the standard baud rates (300, 600, 1200, 2400, 4800, 9600 baud). The timer is programmed with the appropriate divisor (Figure 1) for the selected baud rate resulting in one pulse on the timer output for each bit cell time. The clock output (CLK) of the 8085A is used to clock the timer (TIMERIN). The frequency of this clock is one-half the crystal frequency or in this example 3.072 MHz. TIMEROUT now provides a crystal controlled pulse train at the baud rate selected.

## Serial Communications

By feeding the TIMEROUT signal of the 8156 back to the edge triggered RST 7.5 input of the 8085A, the processor can be interrupt driven at

the required baud rate. As shown in Figure 1, the minimum system supports serial communications with only the addition of the send and receive interface circuits.

The SID (SERIAL INPUT DATA) line and the SOD (SERIAL OUTPUT DATA) line are connected directly to a TTY or RS232 interface circuit. Assuming inverted data at the SID input, a direct connection is made to the RST6.5 input for detection of the start bit.

Additional insight into using the 8085's serial I/O lines in communications application can be found in Section 2 of this Appendix.

BAUD RATE	COUNT (DECIMAL)
300	10,240
600	5,120
1200	2,560
2400	1,280
4800	640
9600	320

FIGURE 1. BAUD RATES



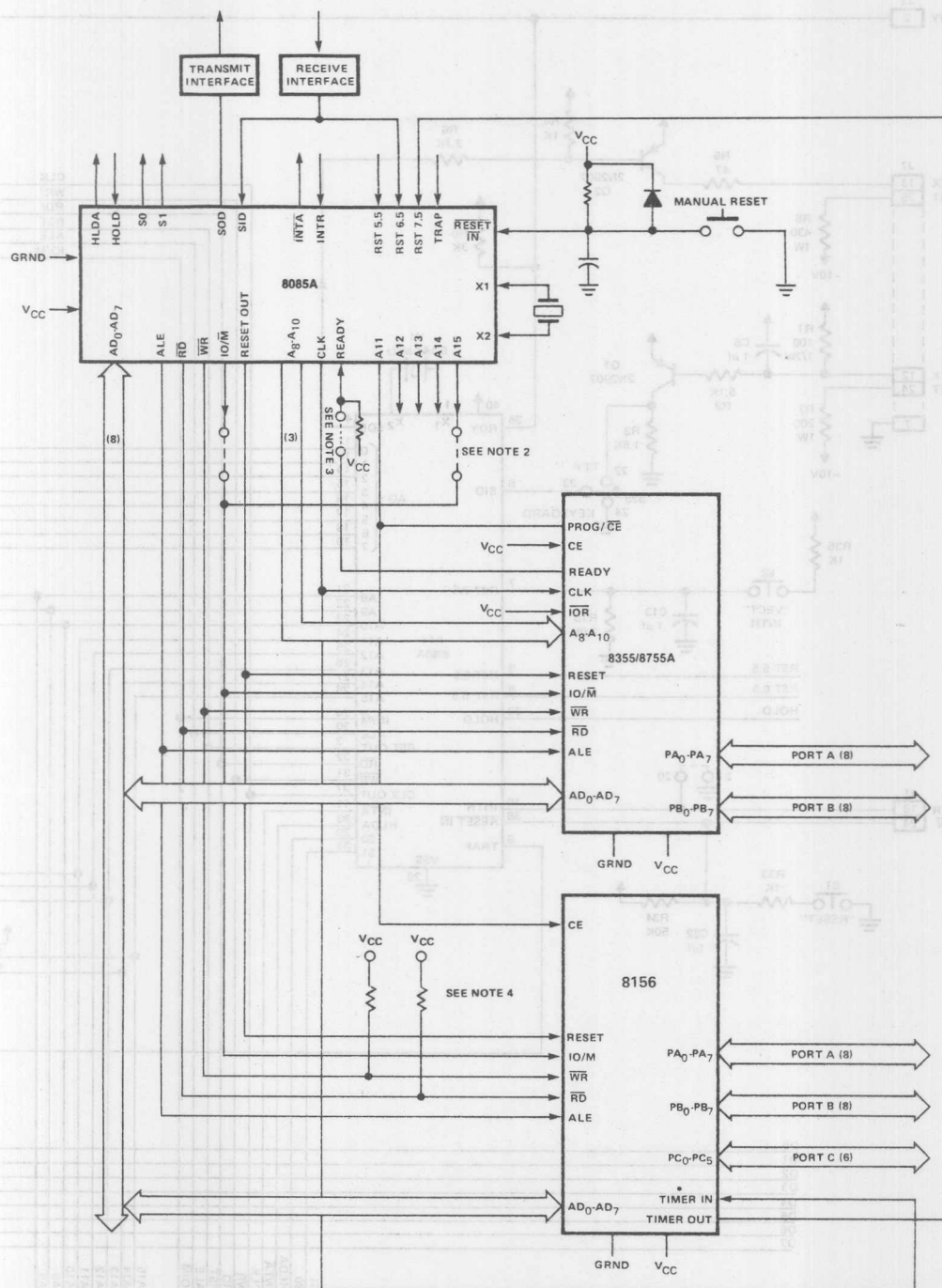
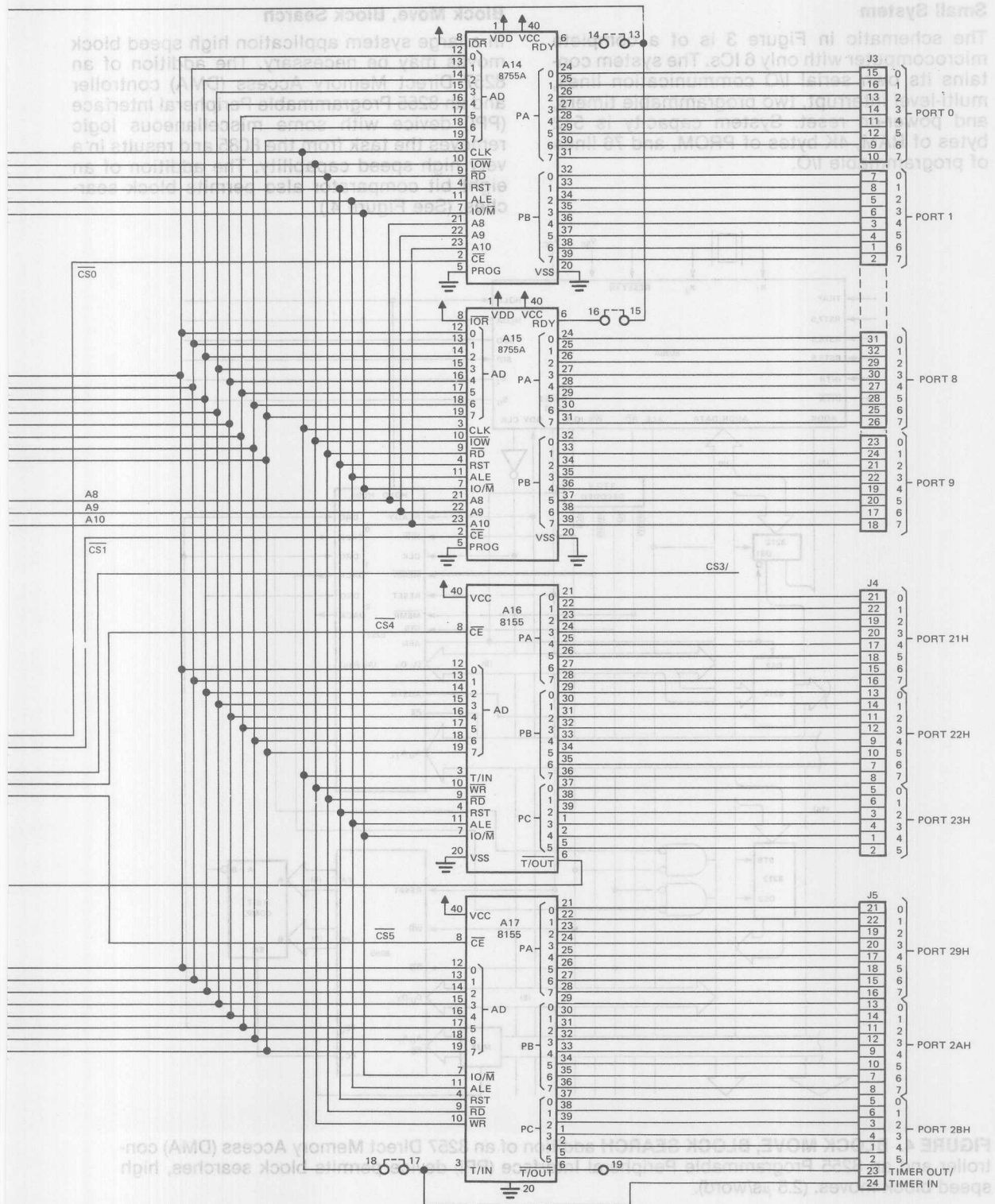


FIGURE 2. MINIMUM SYSTEM CONFIGURATION



## MCS-85™ APPLICATIONS

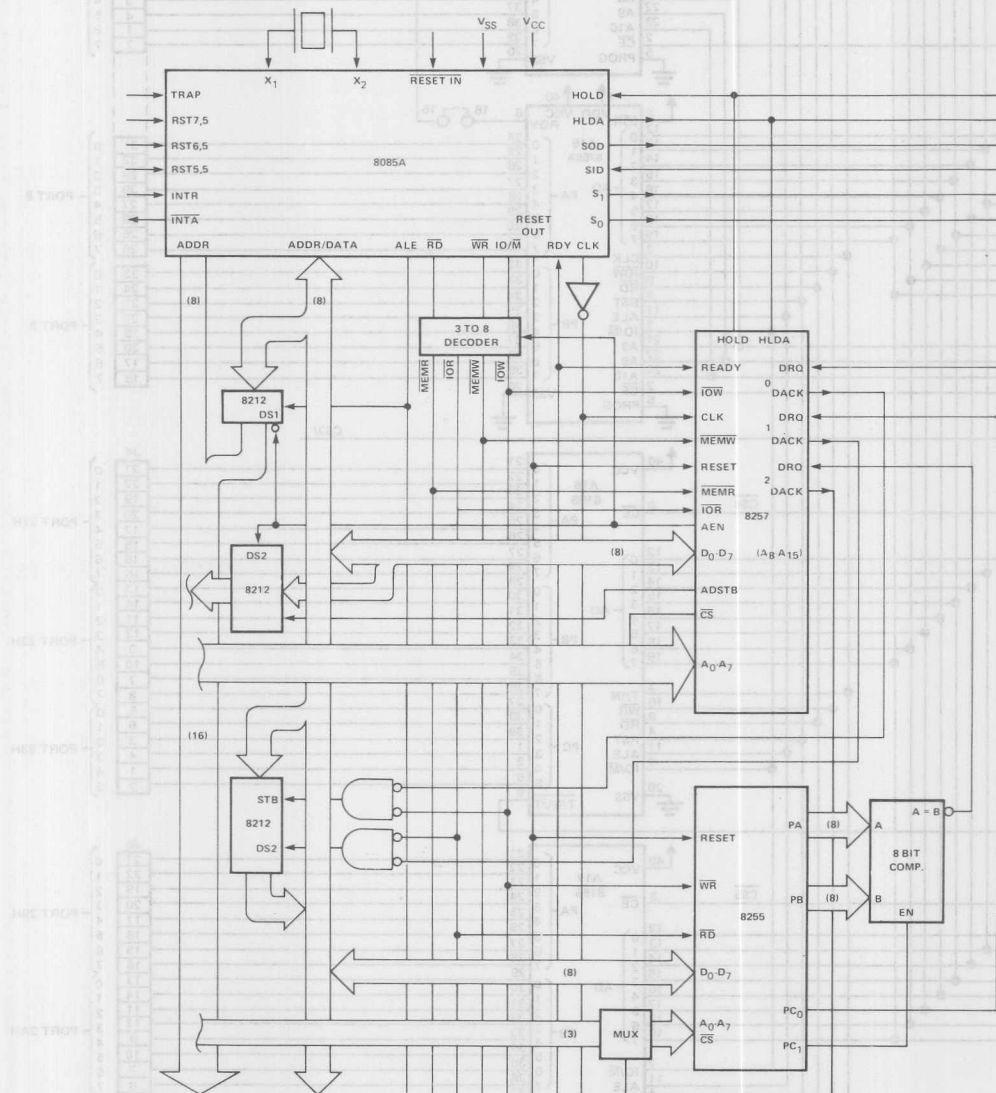


### Small System

The schematic in Figure 3 is of a complete microcomputer with only 6 ICs. The system contains its own serial I/O communication lines, multi-level interrupt, two programmable timers, and power-on reset. System capacity is 512 bytes of RAM, 4K bytes of PROM, and 76 lines of programmable I/O.

### Block Move, Block Search

In a large system application high speed block moves may be necessary. The addition of an 8257 Direct Memory Access (DMA) controller and an 8255 Programmable Peripheral Interface (PPI) device with some miscellaneous logic removes the task from the 8085 and results in a very high speed capability. The addition of an eight bit comparator also permits block searches. (See Figure 4.)



**FIGURE 4. BLOCK MOVE, BLOCK SEARCH** addition of an 8257 Direct Memory Access (DMA) controller and an 8255 Programmable Peripheral Interface (PPI) device permits block searches, high speed block moves. (2.5  $\mu$ s/word).

Basic operation, for a block move, is that the CPU loads the 8257 with the starting address of the source block and the length\* of the block into Channel 0. Channel 1 is programmed with the starting location of the destination block and the length. A bit in Port C of the 8255 is set by the CPU which initiates a DMA request on Channels 0 and 1. Because the 8257 is initialized to the rotating priority mode, the first DMA cycle is from Channel 0 which latches the data from the first location of the source block into the 8212. The second cycle will be from Channel 1 which will store the latched data into the first location of the destination block. The next cycle will return to Channel 0 and the sequence will start over again until the length (terminal count) is reached. Programming the 8257 stop bit insures that each channel will be disabled when its respective terminal count is reached.

This configuration also supports a block fill. DMA Channel 0 points to a location containing the fill value and has a length of one. Channel 1 points to the starting location of the destination block and contains the length. When the sequence is initiated the value will be loaded into the latch by Channel 0. Channel 0 reaches TC and is disabled. Priority rotates to Channel 1 which will repeatedly write into the destination block the value stored in the latch until TC is reached.

Block search operations use the 8-bit comparator and Ports A & B of the 8255 and Channel 2 of the 8257. The CPU loads Port B with the search value and the DMA channel with the search area (starting address and length). A Port C bit initiates the DMA READ request. Channel 2 DMA Acknowledge sets Port A of the 8255 up as the receiver for the DMA READ cycle by multiplexing A<sub>0</sub>, A<sub>1</sub>, and CS. Each cycle of the DMA then loads Port A with the value of the

pointed-to location in the block. When Port A equals Port B, the output of the comparator will gate off the DMA request. The requesting program can now read the Channel 2 address which is pointing to the search value plus one. However, if the status register of the 8257 indicates that TC of Channel 2 has been reached, then no match was found.

## RST 7

On the 8080A/8228 system if one tied INTA out of the 8228 to +12 volts through a 1KΩ resistor, the 8228 would generate a RST 7 instruction to the 8080A upon interrupt. This was a very inexpensive mechanism.

The 8085A has expanded this facility with the RST 5.5, 6.5, 7.5 inputs but is not compatible with the RST 7 generated by the 8228. (Figure 5) To maintain this compatibility it can be achieved by adding an 8212 which will force a RST 7 instruction into the bus upon interrupt acknowledge (INTA). (Figure 6)

RESTART	VECTOR LOCATION
RST 7	38 <sub>16</sub>
RST 5.5	2C <sub>16</sub>
RST 6.5	34 <sub>16</sub>
RST 7.5	3C <sub>16</sub>
TRAP	24 <sub>16</sub>

FIGURE 5. ADDITIONAL 8085A INTERRUPTS

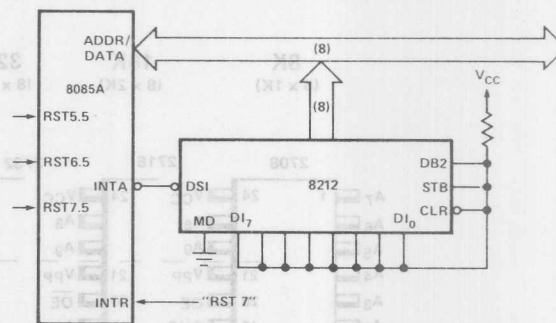


FIGURE 6. 8085A "RST 7" IMPLEMENTATION

\*The value loaded into the low-order 14-bits of the terminal count register specifies the number of DMA cycles minus one before the Terminal Count (TC) output is activated. For instance, a terminal count of 0 would cause the TC output to be active in the first DMA cycle for that channel. In general, if Length = the number of desired DMA cycles, load the value Length-1 into the low-order 14-bits of the terminal count register.)



## Memory Addressing

One of the necessary functions of the microprocessor bus is to interface with the memory where the program is stored. ROM and EPROM memories are typically used to store programs while static and dynamic RAMS are generally used for data memory. The following discussions cover the interfacing to be used for these types of memory.

### ROM - EPROM ADDRESSING

Later in this Appendix a section is devoted to an approach for developing a chart showing memory device compatibility for the 8085A. However, there is one area not included that will be discussed here, that is, unbuffered interfacing to standard ROM or EPROM memories. To use an unbuffered interface to ROM or EPROM it is necessary to understand a particular characteristic of the 8085A.

The 8085A has a period of time, T4 through T6 of the op code fetch cycle and certain instructions, where addresses A8 through A15 are undefined. Be careful about this. Not having addresses stable and using an address select method that would randomly turn on memory devices will cause bus contention and reliability problems in the unbuffered system. In the memory compatibility section of this Application Note, a minimum (unbuffered MCS-85 family and medium system (at least one level of buffering) configurations are considered. These configurations do not have bus contention problems. In the minimum system only MCS-85 components will be discussed where addresses are latched on the falling edge of ALE, thus ignoring any extraneous address transitions. The medium system is assumed to have data buffers that are enabled only at the proper time, thus again preventing any bus contention problems. What about the user who wants to use standard ROM or EPROM without buffering?

As an example let's look at Intel's ROM/EPROM family (Fig. 7) and develop a system block diagram. This system should allow upward compatibility for these particular devices and avoid any bus contentions due to undefined addresses. In Figure 8 a traditional decoding scheme is shown that uses the time difference between t<sub>acc</sub> (address access) and t<sub>co</sub> (chip select access) to allow for decoding of the EPROM/ROM to be selected. Connecting only these signals, however, in an unbuffered system will result in data contention because of the spurious addresses during opcode fetch. The proper interconnect for this type of interface is shown in Figure 9 where an output enable ( $\overline{OE}$ ) signal will prevent any bus contention. This output enable is controlled by the read control signal,  $\overline{RD}$ , of the 8085A. This signal only occurs after addresses have stabilized.\*

Note also that a PROM is recommended for the decoding function vs. an 8205 (1 of 8 decoder). Why? This PROM allows the user to easily upgrade his system to the 32 and 64K versions with minimum rewiring. As seen in Figure 3, only 4 pins are being altered (18-21) in the Intel ROM/EPROM family to allow for this upward compatibility. All a user would need to do is initially design his layout for 28 pin devices, thereby allowing total flexibility from 8K through 64K with the ease of only changing a decoding PROM and a few wires.† Application Note AP-30 can be ordered at no charge which fully discusses the application of Intel's 5 Volt EPROM and ROM family for microprocessor systems.

\*Both  $\overline{RD}$  and  $\overline{WR}$  signals should be pulled up to +5V through a resistor to avoid random selection during 3-state.

†Another method is shown later in Figure 15 that facilitates the use of a decoder, such as the Intel 8205.

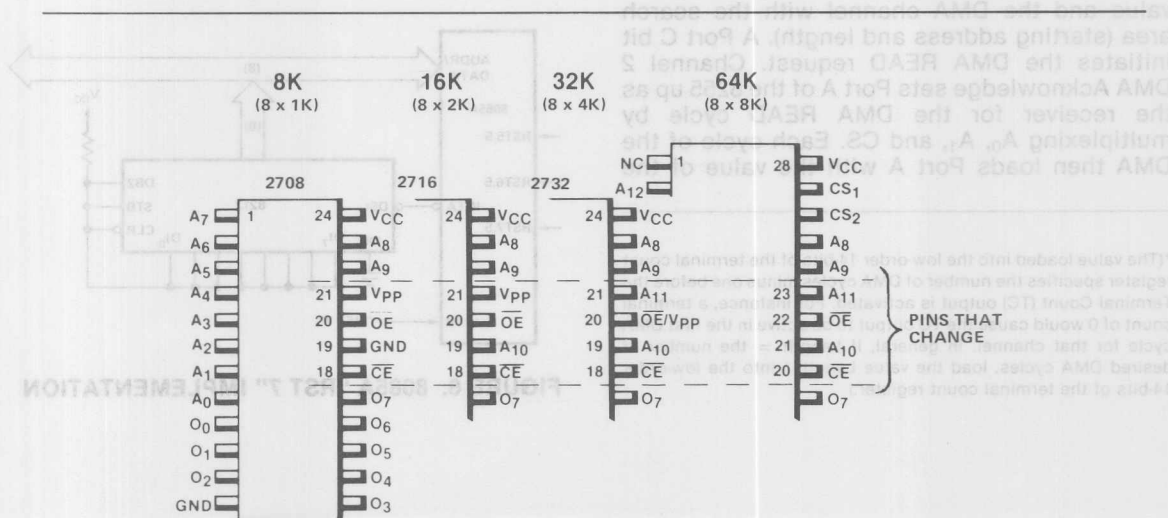
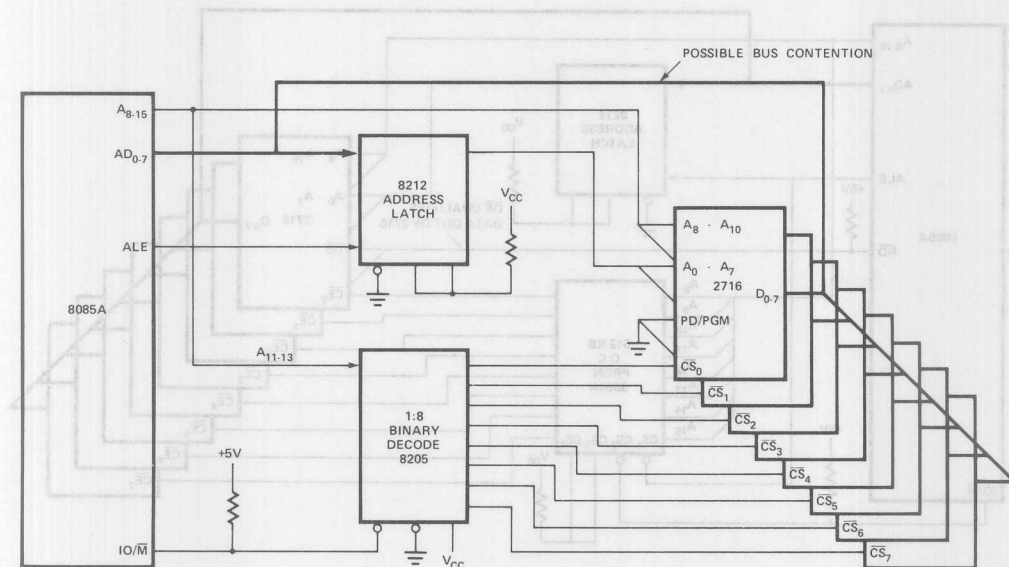


Figure 7. Intel® EPROM/ROM Compatible Family





## STATIC MEMORIES

The same consideration must be applied to standard static memories as with the ROMs/EPROMs in an unbuffered system. Memory device selection must be qualified by a memory read or write to prevent spurious selection. Some Intel static RAM devices have an Output Enable for this purpose, such as the 2142 (1k x 4). This part was designed to be specifically used with a microprocessor bus. For other standard static RAMs, the chip selects must be qualified by  $\overline{RD}$ ,  $\overline{WR}$  or ALE to prevent random selection.

## DYNAMIC RAM INTERFACE

An earlier Intel Application Report (APR-1) extensively covered dynamic RAM interface with different types of memory and refresh in the MCS-80 system. This dynamic RAM section was taken from the most memory intensive example in APR-1, the 2116, modified to be compatible with the 8085A bus. These minor modifications are such that an 8080 system can be converted without much trouble. Before discussion of this section, however, a strong word of advice is in order. At about the same time this Application Note is published, Intel will be sampling an 8202 dynamic RAM refresh controller which does all dynamic RAM interfacing (except the data bus) and refreshing in one packaged component. It is highly recommended that the reader investigate this before using the attached schematic. Reading this section will still be useful in terms of understanding the 8085A bus.

This section uses the APR-1 2116 (multiplexed address 16K) example modified for the 2117-4 dynamic RAM. These devices have some differences from the 2116. One is that the output is not latched and is 3-stated during a write operation. This allows a user to tie both the data in and data out pins together at the device and at the data buffers, saving board traces. The 2117 also have hidden refresh capabilities where if  $\overline{CAS}$  is held low,  $\overline{RAS}$  can be toggled to refresh the device.

The schematic shown in Figure 10 is aimed at a high performance, relatively inexpensive solution (disregarding the 8202). Refresh circuitry is not shown, but can be implemented in a variety of ways. This will be discussed later in an upcoming section. In this refresh section, code for a simple, very low cost refresh controller that requires no special hardware, other than an 8155 timer, is presented.

For system timing, a 4x clock is used to obtain the resolution necessary to provide the clocks for the multiplexed address 2117's. Other solutions are possible with delay lines, one shots, etc., but are relatively expensive and don't provide for a nice baud rate source for any peripherals that may be in the system as does this 4x clock. Another approach can use the clock edges from the 8085A CLKOUT to interface to dynamic RAM. To facilitate this type of approach, Clock related timing parameters are listed later in this note.

To aid in understanding the operation of this circuit, the explanation is broken into a discussion of the main signal paths. 2117-4 Spec compatibility with the 8085A will be discussed in detail in the dynamic RAM section of the Memory Compatibility section.

## Addresses

The lower 14 addresses (A0-A13) are used to select one of the 16,384 8-bit bytes in each 16K byte data bank. The lower 8 of these 14 addresses (A0-A7) flow through an 8212 and are latched by ALE, effectively demultiplexing the address/data bus. These lower 8 addresses with the next 6 (A8-A13) enter the 3242 multiplexer/refresh controller. The Row Enable of the 3242 controls which half of the addresses are presented to the dynamic RAM memory. Looking at the row enable on the 3242, it is seen that the row and column addresses are swapped with respect to convention. The higher order addresses are used as row addresses and the lower order addresses are used as column addresses. This does not create problems because this is invisible to the CPU. Refreshing is done properly as the 3242 controls the addressing for this. The upper two address lines (A14-A15) are decoded to qualify one of the four  $\overline{RAS}$  (Row Address Strobe) lines to select one of the four 16K byte data banks of memory.

## Cycle Requests

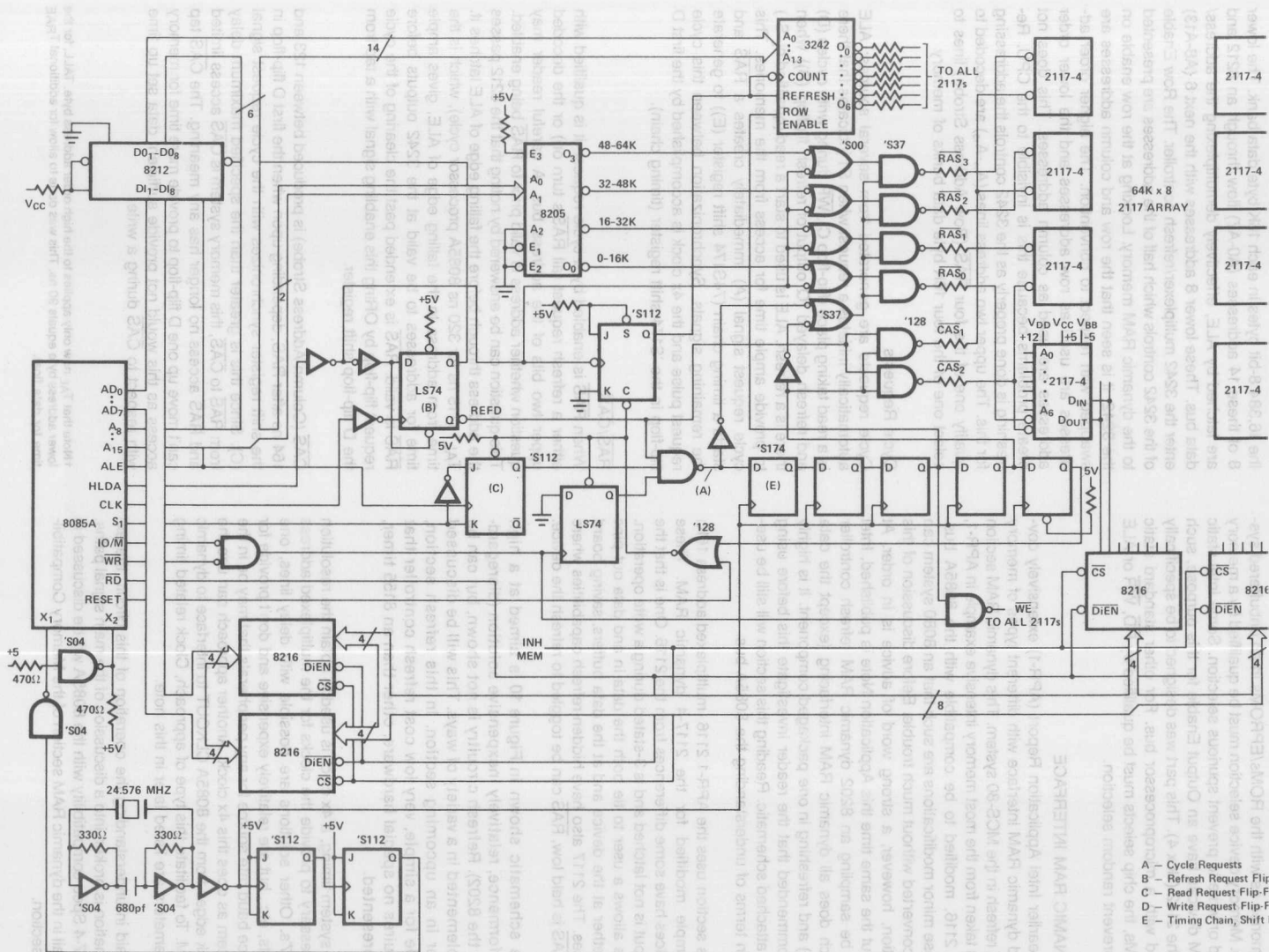
Cycle requests are generated from several sources; ALE automatically initiates a request when S1 indicates that there is a read taking place (flip-flop C),  $\overline{WR}$  during write cycles (D) and refresh delayed (Q output of refresh flipflop (B)) when there is a refresh. ALE is used to start a read (qualified by S1) to provide ample time for access from the memories. This cycle request signal (A) immediately creates a  $\overline{RAS}$  and starts a timing chain (74S174 shift register (E)) to generate the remaining signals. Synchronization between this cycle request pulse and the 4x clock is accomplished by the first D flip-flop in the 'S174 shift register (timing chain).

## $\overline{RAS}/\overline{CAS}$

When  $\overline{RAS}$  is enabled by a cycle request, it is qualified with either a refresh request (all  $\overline{RAS}$ 's turn on) or the decoded upper two bits of the address bus. A careful reader may question whether address is valid prior to  $\overline{RAS}$  being enabled. This question can be answered by noting that the 8212 passes the address through before the falling edge of ALE latches it.  $T_{AL}^{\dagger}$  (115 ns for 320 ns 8085A processor cycle), which is the time from address to the falling edge of ALE, gives ample time for addresses to be valid at the 3242 outputs before  $\overline{RAS}$  is valid.  $\overline{RAS}$  is extended past the clearing of the cycle request flip-flop by ORing this enabling signal with a tap from the D flip-flop shift register.

$\overline{CAS}$  (Column Address Strobe) is produced between 123 and 164 ns after  $\overline{RAS}$ , depending upon when the first D flip-flop in the shift register synchronizes with the cycle request signal (C). Since this is greater than the specified maximum delay from  $\overline{RAS}$  to  $\overline{CAS}$ , this memory system is  $\overline{CAS}$  access limited and  $\overline{RAS}$  access no longer has any meaning. The  $\overline{CAS}$  tap can't move up one D flip-flop to provide more time for memory access as this would not provide sufficient data set up time with respect to  $\overline{CAS}$  during a write.

<sup>†</sup>Note that  $T_{AL}$  now only applies to the high order address byte.  $T_{ALL}$ , for the lower address byte equals 90 ns. This was done to allow for additional  $T_{RAE}$  time for data float.





## Data

The data path to the 2117s is through two sets of buffers to account for memory being off board. To determine bus timing it is helpful to know that Write data is not guaranteed to be valid from the 8085A until 40 ns after the leading edge of the write control signal. On account of this and the delay times for the buffers it is necessary to delay the cycle request on a write until the WR signal goes low. The solution shown still does not require wait states. An inhibit memory signal is also involved. This is useful when using memory address space overlap such as the case with bootstrap ROM (which would be necessary in this system if a full 64K of dynamic RAM is used).

## Refresh

Dynamic RAMs are generally refreshed in two different modes; burst (i.e., all at once every 2 ms) and distributed (one row every (2 ms/number of rows) period of time). The schematic shown provides for a distributed refresh where refresh requests are applied to the Hold request input of the 8085A (not shown). This signal needs to occur at least once every 15  $\mu$ sec ((2ms/128 rows to be refreshed) - HOLD to HLDA delay) and can be generated through a baud rate timing chain, Intel 3222, one shots or other similar devices. Another approach to refresh could qualify the refresh cycles with program fetch cycles (use status lines). If program memory is in static RAM or ROM and the dynamic RAM bus can be isolated, refresh cycles can be performed with no overhead. Instead of using the HOLD feature of the 8085A, refresh can be hidden in the program fetch and decode. Further considerations for refresh include proper handling of resets and excessive hold times from other peripherals to be certain the memory is being refreshed adequately.

Some applications don't require high CPU efficiency and require a very inexpensive method to refresh their dynamic RAM. Since writing, reading or performing special refresh cycles all refresh a particular row, why not do "dummy" reads to refresh? To use this technique memory must be mapped on a one to one correspondence with the address space. This will allow the programmer to read one byte in each physical row in the 2117s, thereby refreshing that row. A simple software routine can be devised to refresh 16K bytes of RAM. If more dynamic RAM than this is desired it can be accomplished by specially enabling all the desired RAS signals via an 8085A output port. First let's analyze how many CPU cycles are available in the 2ms period:

$2\text{ms}/(320 \text{ ns/cycle}) = 6,250 \text{ cycles}$   
for 8085A @ 3.125 MHz

$2\text{ms}/(200 \text{ ns/cycle}) = 10,000 \text{ cycles}$   
for 8085A-2 @ 5.0 MHz

If there is a convenient component that can count 8085A cycles (8085A CLKOUT) and interrupt the 8085A, you're home free. An example of such a device is the 8155 in the MCS-85 family. On the 8155 one can use the  $\overline{\text{TO}}$  (timer out) pin to interrupt the CPU everytime a refresh needs to be performed and an interrupt service routine could dummy read 128 consecutive locations and return to CPU operation. (128 reads are necessary to completely refresh the full 16K bytes of 2117 memory.) The highest priority interrupt should be used for this to insure that refresh occurs. Figure 11 is an example program to perform this burst dummy read refresh. This routine basically uses 64 pops of the stack, each reading two consecutive locations in the memory. Note that this routine destroys the contents of registers B, C and D in the 8085A. The user may want to save these registers in the routine before performing the software refresh. If memory space is more valuable than CPU efficiency, the POPs can be performed in a loop instead of a string, saving additional memory.

This routine requires 690 cycles which is about 11% of the available 8085A CPU cycles, or 7% of the available 8085A-2 cycles. If this is acceptable and there is a counter available, you can't find a cheaper way to do refresh. Note that as processor speeds become faster, this overhead becomes proportionately less and more attractive as an alternative. Again, as with any refresh routine, reset and excessive holds must be dealt with to guarantee proper refresh.

## DMA (Direct Memory Access)

DMA is becoming more common in the microcomputer system for many applications. Some examples include the 8271 floppy disk controller and refreshing a CRT via an 8275 CRT Controller. It is always helpful to reduce the overhead of the DMA (as DMA can tie up the system bus) whenever possible. In many applications, where program memory is resident in ROM or PROM, DMA cycles can be performed in coincidence with op code fetch. This will make them invisible to the CPU as described for Refresh in the Refresh section of the 2117 dynamic RAM example.

In the dynamic Ram system, Refresh requests can be made on the DMA controller via the DRQ lines, with the 8237 in a rotating priority mode to insure refreshing is done. Another technique would be to devise an arbiter for DMA and refresh requests at the processor hold input. With this technique the designer must not allow DMA to monopolize the bus when refresh is needed.

During initialization:

```

MVI A, D5H SET TIMER COUNT TO 5550* FOR REFRESH COUNT
OUT TIMER MSBYTE
MVI A, A4H INTERRUPT CPU AT T0 (TIMER OUT)
OUT TIMER LSBYTE
MVI A, C0H START COUNTER, PLACE C0 IN 8155 STATUS REG.
OUT TIMER COMMAND

```

Program

```

AT RST 7.5 RETURN ADDRESS CALL RFRS (REFRESH SERVICE)
TOTAL #CYCLES
CYCLES
10 RFRS: LXI HL, 0 SAVE STACK POINTER IN HL
10 DAD SP
30 LXI SP, 0080 32K - 48K REFRESH
10 POP BC
10 POP BC REFRESH, DUMMY READ
64 TIMES
640 SPHL RESTORE STACK POINTER
4 EI ENABLE INTERRUPTS
20 RET RETURN
690 TOTAL CYCLES (round up to 700)

```

Figure 11. Software Refresh

The standard technique for interfacing the 8085A processor to the 8237 DMA controller is shown in the MCS-85 User's Manual and is reproduced in Figure 12. This configuration is set up to interface with standard memories or peripherals, i.e., ones that don't share their data bus with addresses, not the MCS-85 family components (8155, 8355, 8755A, etc.). DMA is unlikely with these MCS-85 components as they are intended for

minimum system applications. If the system has both MCS-85 and standard addressed components, and DMA is used for the standard addressed components, ALE must be or'ed with ADSTB from the 8257. This is necessary to deselect the MCS-85 components from the bus. Due to the latching feature of the MCS-85 components, bus contention may result if this is not done and DMA tries to use the bus.

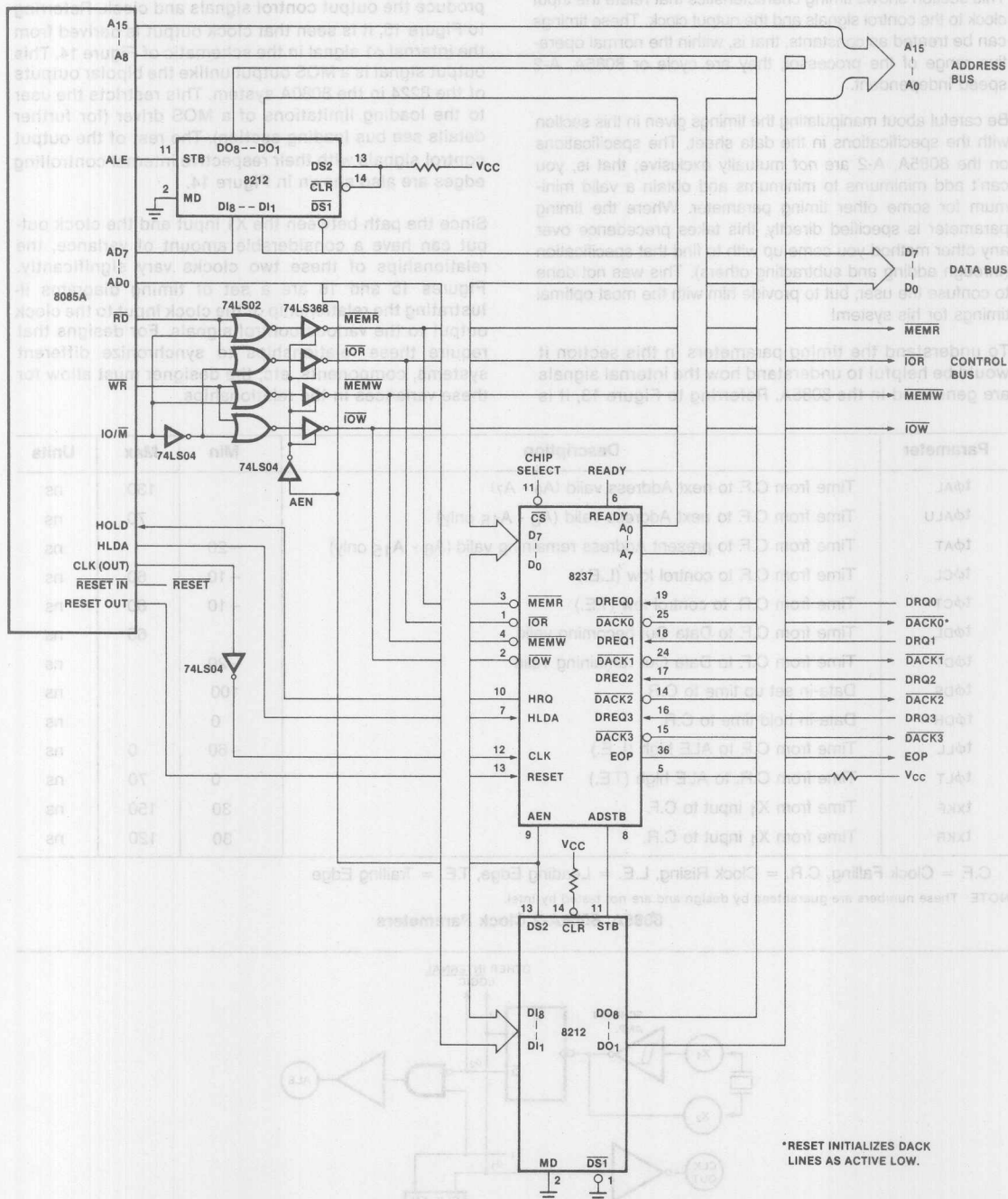


Figure 12. Detailed System Interface Schematic

## SYSTEM TIMINGS

### 8085A CLK-IN vs. CLK-OUT vs. Control Timings

This section shows timing characteristics that relate the input clock to the control signals and the output clock. These timings can be treated as constants, that is, within the normal operative range of the processor, they are cycle or 8085A, A-2 speed independent.

Be careful about manipulating the timings given in this section with the specifications in the data sheet. The specifications on the 8085A, A-2 are *not* mutually exclusive; that is, you can't add minimums to minimums and obtain a valid minimum for some other timing parameter. Where the timing parameter is specified directly, this takes precedence over any other method you come up with to find that specification (through adding and subtracting others). This was not done to confuse the user, but to provide him with the most optimal timings for his system!

To understand the timing parameters in this section it would be helpful to understand how the internal signals are generated in the 8085A. Referring to Figure 13, it is

seen that the rising edge of the X1 input causes flip-flop A to toggle. From this flip-flop two internal signals are generated that drive all functions in the 8085A, A-2 and produce the output control signals and clock. Referring to Figure 15, it is seen that clock output is derived from the internal  $\phi_1$  signal in the schematic of Figure 14. This output signal is a MOS output unlike the bipolar outputs of the 8224 in the 8080A system. This restricts the user to the loading limitations of a MOS driver (for further details see bus loading section). The rest of the output control signals with their respective internal controlling edges are also shown in Figure 14.

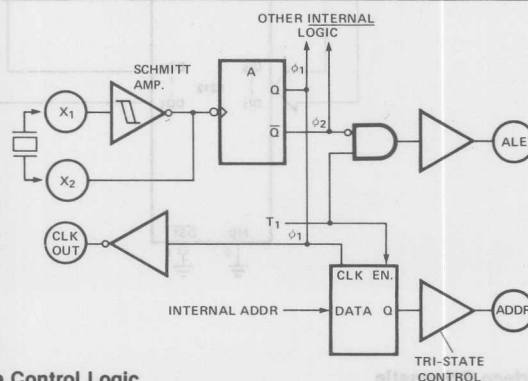
Since the path between the X1 input and the clock output can have a considerable amount of variance, the relationships of these two clocks vary significantly. Figures 15 and 16 are a set of timing diagrams illustrating the relationship of the clock input to the clock output to the various control signals. For designs that require these relationships to synchronize different systems, components, etc; the designer must allow for these variances in the relationships.

Parameter	Description	Min	Max	Units
t $\phi$ AL	Time from C.F. to next Address valid (A <sub>0</sub> - A <sub>7</sub> )		130	ns
t $\phi$ ALU	Time from C.F. to next Address valid (A <sub>8</sub> - A <sub>15</sub> only)		70	ns
t $\phi$ AT	Time from C.F. to present Address remaining valid (A <sub>8</sub> - A <sub>15</sub> only)	-20		ns
t $\phi$ CL	Time from C.F. to control low (L.E.)	-10	60	ns
t $\phi$ CT	Time from C.R. to control low (T.E.)	-10	60	ns
t $\phi$ DL	Time from C.F. to Data Out becoming valid		65	ns
t $\phi$ DT	Time from C.F. to Data Out remaining valid	-20		ns
t $\phi$ DS	Data-in set up time to C.R.	100		ns
t $\phi$ DH	Data-in hold time to C.R.	0		ns
t $\phi$ LL	Time from C.F. to ALE high (L.E.)	-60	0	ns
t $\phi$ LT	Time from C.R. to ALE high (T.E.)	0	70	ns
t $\chi$ KF	Time from X <sub>1</sub> input to C.F.	30	150	ns
t $\chi$ KR	Time from X <sub>1</sub> input to C.R.	30	120	ns

C.F. = Clock Falling, C.R. = Clock Rising, L.E. = Leading Edge, T.E. = Trailing Edge

NOTE: These numbers are guaranteed by design and are not tested by Intel.

### 8085A, 8085A-2 Clock Parameters



**Figure 13. Clock and Sample Control Logic**

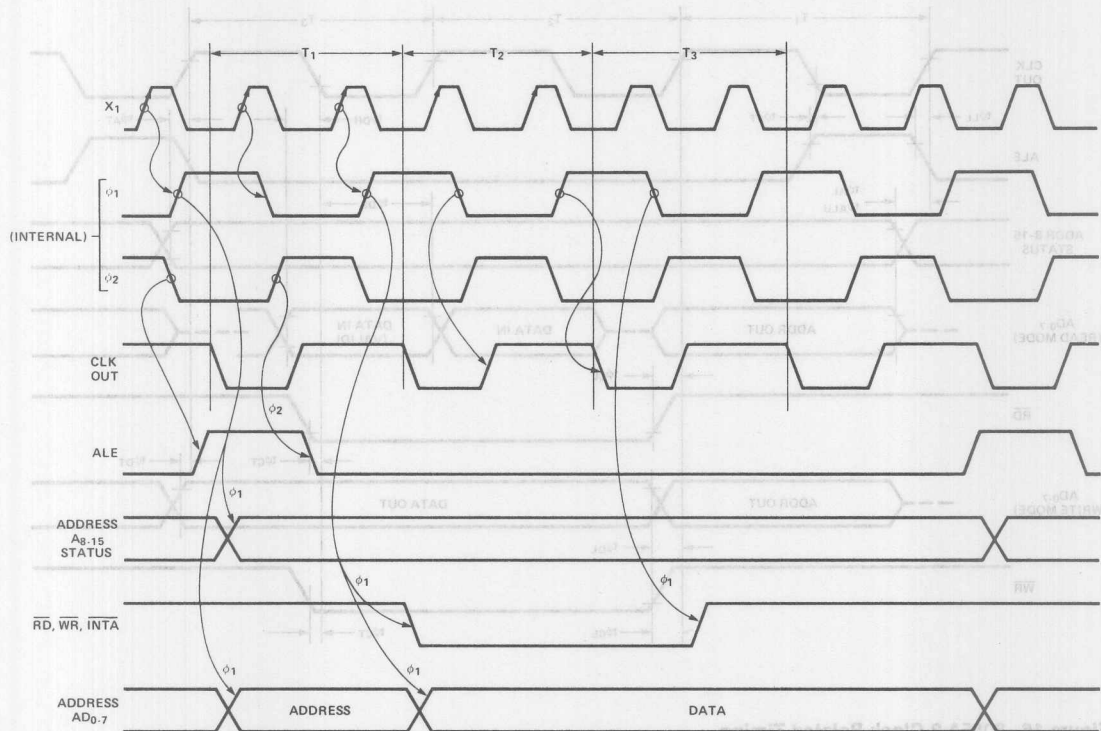


Figure 14. Clock In (X1) to Output Relationship

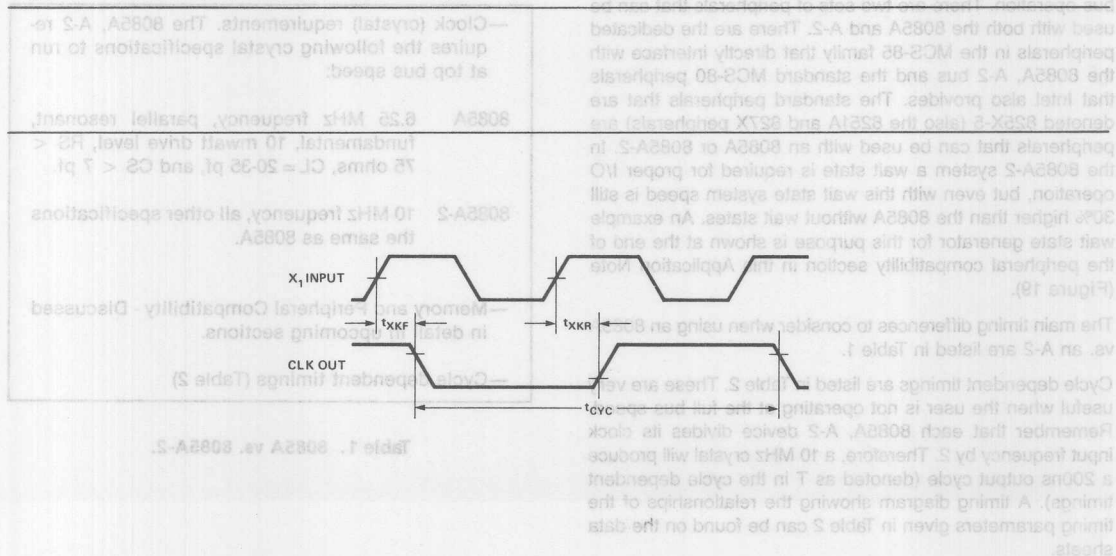


Figure 15. 8085A-2 Clock In/Clock Out Timing



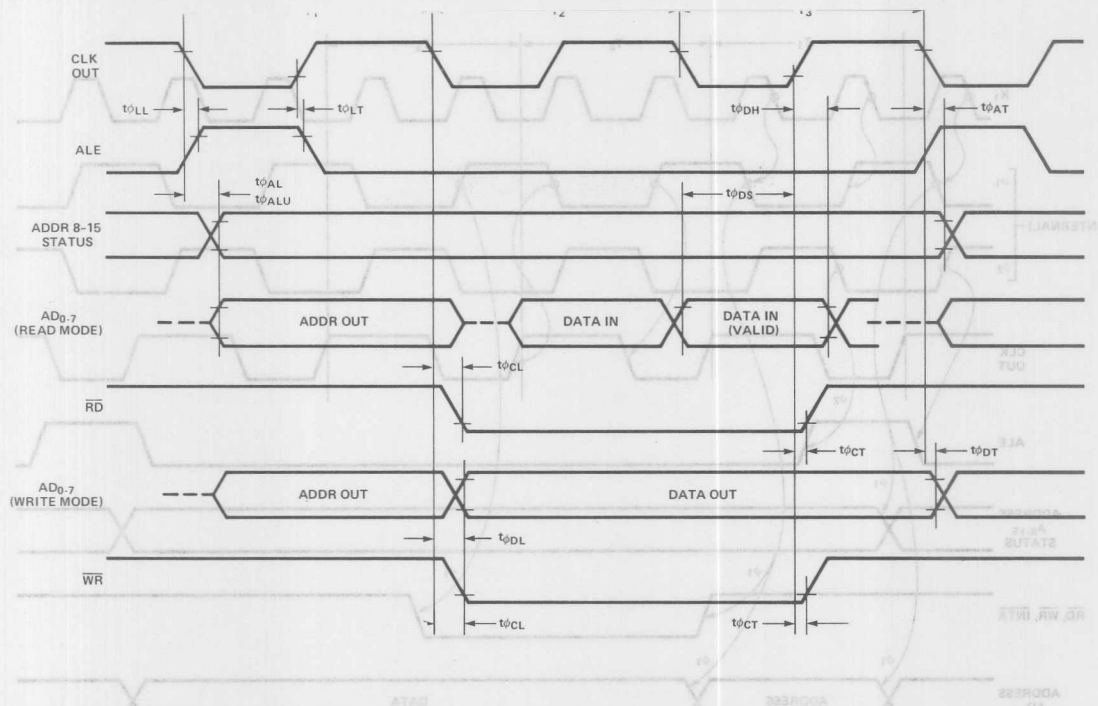


Figure 16. 8085A-2 Clock Related Timing

### 3.125 vs. 5 MHz Considerations

The 8085A (with maximum internal clock frequency of 3.125 MHz) and 8085A-2 (5 MHz) have some differences in their bus operation. There are two sets of peripherals that can be used with both the 8085A and A-2. There are the dedicated peripherals in the MCS-85 family that directly interface with the 8085A, A-2 bus and the standard MCS-80 peripherals that Intel also provides. The standard peripherals that are denoted 825X-5 (also the 8251A and 827X peripherals) are peripherals that can be used with an 8085A or 8085A-2. In the 8085A-2 system a wait state is required for proper I/O operation, but even with this wait state system speed is still 30% higher than the 8085A without wait states. An example wait state generator for this purpose is shown at the end of the peripheral compatibility section in this Application Note (Figure 19).

The main timing differences to consider when using an 8085A vs. an A-2 are listed in Table 1.

Cycle dependent timings are listed in Table 2. These are very useful when the user is not operating at the full bus speed. Remember that each 8085A, A-2 device divides its clock input frequency by 2. Therefore, a 10 MHz crystal will produce a 200ns output cycle (denoted as T in the cycle dependent timings). A timing diagram showing the relationships of the timing parameters given in Table 2 can be found on the data sheets.

—Clock (crystal) requirements. The 8085A, A-2 requires the following crystal specifications to run at top bus speed:

8085A	6.25 MHz frequency, parallel resonant, fundamental, 10 mwatt drive level, RS < 75 ohms, CL = 20-35 pf, and CS < 7 pf.
8085A-2	10 MHz frequency, all other specifications the same as 8085A.

—Memory and Peripheral Compatibility - Discussed in detail in upcoming sections.

—Cycle dependent timings (Table 2)

Table 1. 8085A vs. 8085A-2.

3.125 MHz (8085A)				5 MHz (8085A-2)			
Parameter	Min	Max	Cycle Dependencies	Min	Max	Cycle Dependencies	
t <sub>CYC</sub>	320	2000		200	2000		
t <sub>1</sub>	80		1/2T-80	40		1/2T-70	
t <sub>2</sub>	120		1/2T-40	70		1/2T-50	
t <sub>r</sub>		30			30		
t <sub>f</sub>		30			30		
t <sub>AL</sub>	115		1/2T-45	50		1/2T-50	
t <sub>LA</sub>	100		1/2T-60	50		1/2T-50	
t <sub>LL</sub>	140		1/2T-20	80		1/2T-20	
t <sub>LCK</sub>	100		1/2T-60	50		1/2T-50	
t <sub>LC</sub>	130		1/2T-30	60		1/2T-40	
t <sub>AFR</sub>		0			0		
t <sub>AD</sub>		575	(5/2+N)T-225		350	(5/2+N)T-150	
t <sub>RD</sub>		300	(3/2+N)T-180		150	(3/2+N)T-150	
t <sub>RDH</sub>		0			0		
t <sub>RAE</sub>	150		1/2T-10	90		1/2T-10	
t <sub>CA</sub>	120		1/2T-40	60		1/2T-40	
t <sub>DW</sub>	420		(3/2+N)T-60	230		(3/2+N)T-70	
t <sub>WD</sub>	100		1/2T-60	60		1/2T-40	
t <sub>CC</sub>	400		(3/2+N)T-80	230		(3/2+N)T-70	
t <sub>CL</sub>	50		1/2T-110	25		1/2T-75	
t <sub>ARY</sub>		220	3/2T-260		100	3/2T-200	
t <sub>RYs</sub>	110			100			
t <sub>RYH</sub>	0			0			
t <sub>HACK</sub>	110		1/2T-50	40			
t <sub>HABE</sub>		210	1/2T+50		150	1/2T+50	
t <sub>RV</sub>	400		3/2T-80	220		3/2T-80	
t <sub>AC</sub>	270		T-50	115		T-85	
t <sub>HDS</sub>	170			120			
t <sub>HDH</sub>	0			0			
t <sub>INS</sub>	360		1/2T+200	150		1/2T+50	
t <sub>INH</sub>	0			0			
t <sub>LDR</sub>		460	2T-180		270	4/2T-130	

Where T = t<sub>CYC</sub> and N = the number of wait states that are incorporated.  
All mathematical operations in Table 2 are performed from left to right, except where qualified with parenthesis.

Table 2. 8085A and 8085A-2 Cycle Dependencies

## Memory Device Compatibility

### Determining What Memory to Select For Your Application

When developing a system which will use sufficient memory to require buffering (see the capacitive loading section to determine when it is needed), it is important to understand how to select the slowest, lowest cost memory and still be compatible with the bus timings with minimum wait states. A generalized procedure has been developed in the following section for determining the memory access needed for different applications and the number of wait states required (if any). In general the amount of time available for accessing the memory can be obtained from the following formula: Available memory access = 8085A access time (from control signal of interest) - Buffering/Decoding delay (to and from memory)

The three main "control" signals of interest which determine memory access are that of  $t_{RD}$  (read to valid data in),  $t_{AD}$  (valid address to valid data in) and  $t_{LDR}$  (address latch enable to valid data in). When dealing with different types of memories, one or more of these signals becomes important.

Even though memory access compatibility is probably one of the most important parameters to consider, as this is directly reflected in the price of the memory, it is not the only parameter that is important. Some of the other major timing considerations are as follows:

**WRITE ENABLE** - Is the write enable signal sufficiently long to guarantee a write?

Is data set up properly with respect to this write to be compatible with the memory's requirements?

Is data held long enough?

**DATA FLOAT** - Does your system have sufficient margin to prevent bus contention?

(i.e., Does the memory let go of the data bus in time for the processor to use it? Remember that the 8085A shares its Data Bus with the lower 8 addresses.)

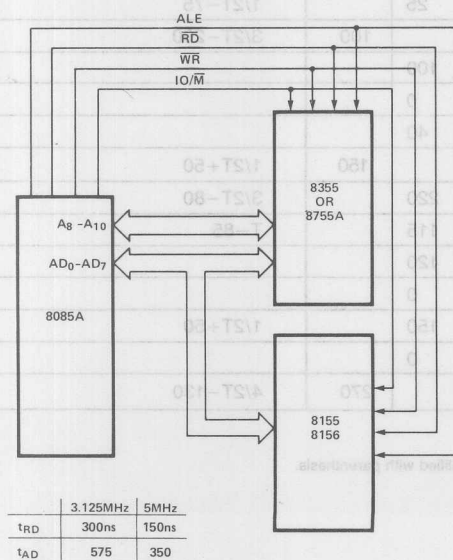


Figure 17. Minimum System

We will first go through the minimum system which can be represented by the dedicated set of components Intel has developed for the 8085A (Fig. 17). The two timing specs were taken from the data catalog for  $t_{RD}$  and  $t_{AD}$  ( $t_{LDR}$  is irrelevant here). Looking at the 8155/6 and 8355/8755A, a comparison can be made for the access times:

	8085A (3.125 MHz)	8155/6	8355/8755A
$t_{RD}$	300 (max)	170 (max)	170 (max)
$t_{AD}$	575 (max)	400 (max)	400/450 (max)

This shows that there is plenty of bus margin for the 3.125 MHz minimum application of the 8085A. Access time for the processor can be interpreted as the time from when the control signal is presented on the bus to the time when the processor will expect the data to be valid so it can sample it. Conversely, memory access times show the amount of time that will elapse between when it is told to present its information to when it actually does it. As long as the memory access spec is less than the processor access spec (minus appropriate buffering delays) the memory is access time compatible.

In more complicated systems where one level of data, address and control buffering is required (such as the case when there are many signal paths and device loading on one card), the delays of the latches and bidirectional drivers must be taken into consideration.

First consider a ROM, EPROM or static RAM configuration as shown in Figure 18. Using the generalized available memory access formula,  $t_{AD}$ ,  $t_{RD}$  and  $t_{LDR}$  for the memory can be determined using the data sheet timing delays for the buffers.

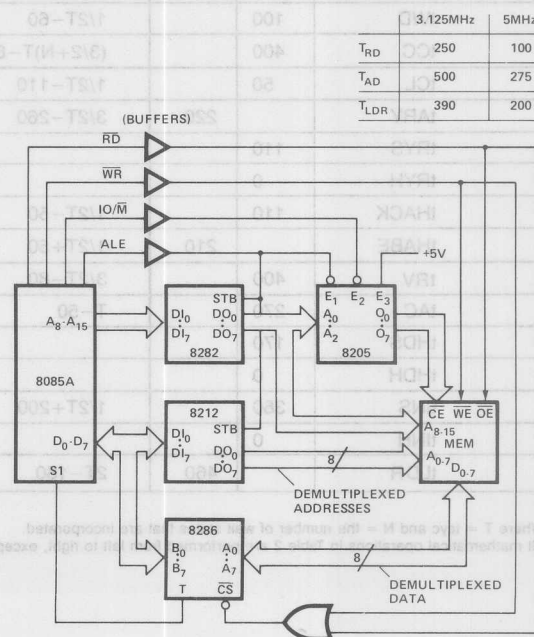


Figure 18. Medium Buffered System

$$\begin{aligned}
t_{AD \text{ MEMORY}} &= t_{AD8085A} - (8282 + 8205 \text{ delay}) - (8286 \text{ delay}) + \text{transitional gain due to buffering}^* \\
&= t_{AD85} - (T_{IVOV} + t_{-}) - (T_{IVOV}) + t_{CAPB}^* \\
&= (5/2+N)T - 225 - 55 - 35 + 15 \\
&= (5/2+N)T - 300 \quad (\text{for } 8085A) \\
&= (5/2+N)T - 225 \quad (\text{for } 8085A-2)
\end{aligned}$$

where N = number of wait states and T = cycle time,

For minimum 8085A timing 500ns =  $t_{AD}$  memory

8085A-2 timing 275ns =  $t_{AD}$  memory

The 8085A timing parameter  $t_{AL}$  was not taken into consideration as the 8282 transfers information directly through without concern of the address latch enable.  $t_{RD}$  can be obtained in a similar manner.

The read signal  $\overline{RD}$  goes through a buffer before it reaches the memory. This must be taken into consideration when calculating effective  $t_{RD}$  for the memory.

$$\begin{aligned}
t_{RD \text{ MEMORY}} &= t_{RD \text{ 8085A}} - (\text{buffer delay}) - (8286 \text{ delay}) + \text{transitional gain due to buffering} \\
&= t_{RD \text{ 85}} - (\text{delay}) - (T_{IVOV}) + t_{CAPB} \\
&= (3/2+N)T - 180 - 30 - 35 + 15 \\
&= (3/2+N)T - 230 \quad (\text{for } 8085A) \\
&= (3/2+N)T - 200 \text{ ns} \quad (\text{for } 8085A-2)
\end{aligned}$$

\* $t_{CAPB}$  is additional time thrown back in for improvement in signal transitions. This is because buffering the signals reduces the capacitive loading considerably. The data sheet gives timings for maximum capacitive loading. Characterization has shown change in delay versus capacitive loading as .12 ns/pF min (under 20 pF loading) and .24 ns/pF max (under 150 pF loading). To take into consideration the effects of this loading two parameters are defined:

$t_{CAPA}$  - delay for a signal to leave the old logic level

$t_{CAPB}$  - delay for a signal to complete the transition from the old to new logic level

where  $t_{CAPA} = 1/2 t_{CAPB}$

	MIN	MAX
$t_{CAPA}$	7 ns	15 ns
$t_{CAPB}$	15 ns	30 ns

In the memory compatibility calculations  $t_{CAPB}$  min is added on as spec sheet values assume 150 pF loading and this system is not worst case, i.e., it has buffering that reduces this loading to approximately 20 pF. Since the CAP = 130 pF and change in delay versus capacitance is 1/2 ns/pF min,  $t_{CAPB \text{ MIN}} = (.1 \text{ ns/pF}) 130 \text{ pF} = \text{approx. } 15 \text{ ns}$ .

For minimum 8085A timing 250ns =  $t_{RD}$  memory

8085A-2 timing 100ns =  $t_{RD}$  memory

Therefore for  $t_{LDR}$ :

$$\begin{aligned}
t_{LDR \text{ MEMORY}} &= t_{LDR \text{ 8085}} - (\text{buffer delay}) - (8205) \\
&= (8286) + t_{CAPB} \\
&= t_{LDR} - (\text{delay}) - (t_{-}) - (T_{IVOV}) + t_{CAPB} \\
&= 2T - 180 - 30 - 20 - 35 + 15 \\
&= 2T - 250 \text{ for } 8085A \\
&= 2T - 200 \text{ for } 8085A-2
\end{aligned}$$

For minimum 8085 timing = 390ns

8085A-2 timing = 200ns

To obtain memory access parameters for a multicard system (which would have buffering at both ends of the system bus), it is a simple matter of subtracting off the additional buffering delays.

With these timings a memory compatibility table can be developed from the data sheets (Table 3). With most of these memories it is relatively straightforward to determine the controlling signal used to select and enable the device. To illustrate this, listed below are the controlling signals of interest for the different memories as they are used in a typical configuration:

Relevant  
Control Signal

RAM  
2114

Address access  
Chip select access

-  $t_{AD \text{ MEM}}$   
-  $t_{LDR \text{ MEM}}^{**}$

2142

Address access  
Chip select access  
Output enable

-  $t_{AD \text{ MEM}}$   
-  $t_{LDR \text{ MEM}}^{**}$   
-  $t_{RD \text{ MEM}}$

ROM

\*\*Chip selects for these static RAMs need not be qualified with ALE. If 2114 or 2142 chip selects are generated directly from the address lines, the relevant timing is  $t_{AP \text{ MEM}}$ .

	3.125 MHz	5 MHz
<b>MINIMUM SYSTEM:</b>		
STATIC RAM	8155/8156, (256x8) 8185 (1Kx8)	8155-2/8156-2 8185-2
ROM/EPROM	8355 (2Kx8) 8755A (2Kx8)	8355-2 8755A-2
<b>BUFFERED SYSTEM:</b>		
STATIC RAM	2114 (1Kx4) 2142 (1Kx4)	2114-2 2142-2
ROM/EPROM	2732 (4Kx8) 2716-2 (2Kx8)	2716-2**
*Contact Intel for high performance EPROM/ROM Family. **With 1 wait state.		

Table 3. 8085A, A-2 Memory Compatibility.

when determining which memory device to use. When there is an output enable, tRD MEM is also used. All relevant access times must be met by the resulting system configuration to be compatible.

This note will not attempt to generalize a procedure that deals with the interface to dynamic RAM, but the 2117 example shown earlier is described below. In the dynamic RAM system, many variables come into play upon which the memory access is dependent. Among these are refresh controllers, decoding, whether or not the system is designed for minimum hardware or maximum performance, and consideration for nonmultiplexed vs. multiplexed address dynamic RAMs.

For the Intel 2107C, which has nonmultiplexed addresses, tAD is the important parameter as it generates the chip selects and chip enables. However, with a multiplexed address part, things are different and both a  $\overline{\text{RAS}}$  and  $\overline{\text{CAS}}$  access time must be considered. Note that since  $\overline{\text{RAS}}$  is applied before  $\overline{\text{CAS}}$ ,  $\overline{\text{RAS}}$  access time is effective only while the  $\overline{\text{CAS}}$  signal stays within the specified  $\overline{\text{RAS}}$  to  $\overline{\text{CAS}}$  delay time. If it is not possible to do this,  $\overline{\text{CAS}}$  access becomes the limiting factor for memory selection. Don't be misled by the  $\overline{\text{RAS}}$  to  $\overline{\text{CAS}}$  maximum delay (tRCD:  $\overline{\text{RAS}}$  to  $\overline{\text{CAS}}$  delay time) spec'd on dynamic RAM data sheets! This maximum only applies to guarantee  $\overline{\text{RAS}}$  access.

For a specific example the following shows how the speed versions were selected for previous 2117 dynamic RAM interface.

READ CYCLE	TAKEN FROM 2117-4 DATA SHEET		DYNAMIC RAM CONFIGURATION	
	MIN	MAX	MIN	MAX
tRAC		250 ns		Doesn't apply
tCAC		165 ns	210 ns	
tREF		2 ms		Not Shown
tRP	150 ns		279 ns	
tCPN	25 ns		472 ns	
tCRP	-20 ns		193 ns	
tRCD	35 ns	65 ns	Outside spec, CAS access limited	
tRSH	165 ns		177 ns	
tCSH	250 ns		300 ns	
tASR	0 ns		55 ns	
tRAH	35 ns		82 ns	
tASC	-10 ns		-4 ns	
tCAH	75 ns		205 ns	
tAR	160 ns		410 ns	
toff	70 ns		See Below*	
tRC	410 ns		720 ns	
tRAS	250 ns		307 ns	
tCAS	165 ns		198 ns	
<p>*There are two parameters that the processor "sees". One is memory access, which has already been covered. The other is when the memory will let go of the bus. To show compatibility here, the following analysis is done:</p> <p>2117 tOFF 70 ns max 8085A tRAE 150 ns min</p> <p>Therefore compatible as <math>\overline{\text{WR}}</math> is used to deselect the 8216's.</p>				

Table 4. Bus Compatibility Analysis (see Figure 11)



WRITE CYCLE	TAKEN FROM 2117-4 DATA SHEET		DYNAMIC RAM CONFIGURATION	
	MIN	MAX	MIN	MAX
t <sub>RC</sub>	410 ns		720 ns	
t <sub>RAS</sub>	250 ns		307 ns	
t <sub>CAS</sub>	165 ns		198 ns	
t <sub>WCS</sub>	-20 ns		34 ns	
t <sub>WCH</sub>	75 ns		164 ns	
t <sub>WCR</sub>	160 ns		287 ns	
t <sub>WP</sub>	75 ns		205 ns	
t <sub>RWL</sub>	100 ns		205 ns	
t <sub>CWL</sub>	100 ns		205 ns	
t <sub>DS</sub>	0 ns		23 ns **	
t <sub>DH</sub>	75 ns		Data held until next cycle	
t <sub>DHR</sub>	160 ns		Data held until next cycle	
**Data is not valid from the 8085A until 40 ns after WR falls.				

**Table 4. Bus Compatibility Analysis (see Figure 11) (Cont'd)**

The numbers in Table 4 were obtained by using the following delay assumptions (Table 5) and very conservative techniques of obtaining minimum 8085A timings. Where no direct specification applied, minimum specs were added assuming 0 ns for any rise or fall times. This is more conservative than necessary. Another approach can be made from the clock related timings discussed in an earlier section.

	DELAY	
	MIN	MAX
Gates	0 ns	7 ns
Flip Flops	0 ns	15 ns
8216s	0 ns	30 ns
D flip flop (Timing Chain)	41 ns	41 ns
3242	0 ns	25 ns (Min 0ns for
8212	0 ns	30 ns synchronization D FF)

**Table 5. Delay Assumptions**

An exhaustive approach as Table 4 will more than pay itself back in terms of debugging the circuit. However, while this analysis may be helpful in understanding an existing circuit, it won't help as much in creating a new one. A general procedure for designing with memories is itemized below:

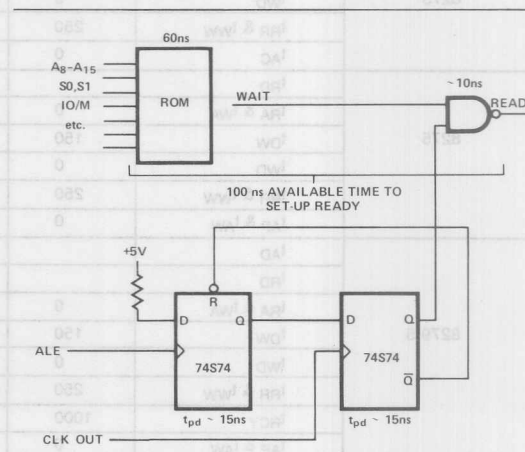
1. Determine how much processor time is available for memory access. Access from addresses is the most important parameter.
2. Determine how much buffering will be used (both to and from the memory) and how much delay there will be due to decode or qualifications in the circuit (in the memory design in Fig. 11, WR qualifies a write). Subtract these resulting delays from step 1 to get an effective access for the memory. If multiplexed address RAM is used go to 3, if not go to 4.
3. Determine how the  $\overline{\text{RAS}}$  and  $\overline{\text{CAS}}$  timings will be generated, be it one shots, delay lines, shift registers, etc. Adjust memory access available for the method chosen.
4. Select a memory that meets this criterion.
5. Design the system to meet all the specified parameters of the memory and verify.

Steps 1, 2 and 4 have been done for you in the Memory Compatibility Table for ROM, EPROM and Static RAM memories in a medium and minimum system. *Remember* - for dynamic RAM, Intel will soon be providing an 8202, a refresh, dynamic RAM controller that generates all  $\overline{\text{RAS}}$ ,  $\overline{\text{CAS}}$  control signals for a 64 kByte memory (made of 2117s).

#### Peripheral Compatibility - 3.125 and 5 MHz

Intel supports its processors with many LSI peripheral components that do a wide range of functions to simplify circuit design. The 8085A compatible peripherals have been denoted the "-5" notation to show compatibility. The "-5" notation also signifies that these devices are compatible with the 8085A-2 with one wait state interjected. This wait state is produced by taking the ready line low at the proper time as shown in Figure 19.

A list of these peripherals is shown in Table 6 with corresponding relevant specifications to illustrate 8085A-2 compatibility. The analysis for determining the resulting timings is similar to the analysis in the previous memory compatibility section.



**Figure 19. 8085A-2 Wait State Generator**

Part No.	AC. Parameter	Min. (ns)	Max. (ns)	8085A-2 AC. Parameter	Margin vs. -2 Spec. (ns)
8251A	t <sub>RD</sub>		200	t <sub>RD</sub>	*150
	t <sub>RA</sub> & t <sub>WA</sub>	0		t <sub>CA</sub>	60
	t <sub>DW</sub>	150		t <sub>DW</sub>	80
	t <sub>WD</sub>	0		t <sub>WD</sub>	60
	t <sub>RR</sub> & t <sub>WW</sub>	250		t <sub>CC</sub>	*180
	t <sub>AR</sub> & t <sub>AW</sub>	0		t <sub>AC</sub>	
8253-5	t <sub>RD</sub>		200	t <sub>RD</sub>	*150
	t <sub>RA</sub>	5		t <sub>CA</sub>	55
	t <sub>WA</sub>	30		t <sub>CA</sub>	60
	t <sub>DW</sub>	250		t <sub>DW</sub>	*180
	t <sub>WD</sub>	30		t <sub>WD</sub>	30
	t <sub>RR</sub> & t <sub>WW</sub>	300		t <sub>CC</sub>	*130
	t <sub>RV</sub>	1000		t <sub>RV</sub>	**
	t <sub>AR</sub> & t <sub>AW</sub>	50		t <sub>AC</sub>	65
8255A-5	t <sub>RD</sub>		200	t <sub>RD</sub>	*150
	t <sub>RA</sub>	0		t <sub>CA</sub>	60
	t <sub>WA</sub>	20		t <sub>CA</sub>	40
	t <sub>DW</sub>	100		t <sub>DW</sub>	130
	t <sub>WD</sub>	30		t <sub>WD</sub>	30
	t <sub>RR</sub> & t <sub>WW</sub>	300		t <sub>CC</sub>	*130
	t <sub>RV</sub>	850		t <sub>RV</sub>	**
	t <sub>AR</sub> & t <sub>AW</sub>	0		t <sub>AC</sub>	115
8257-5	t <sub>RD</sub>		200	t <sub>RD</sub>	*150
	t <sub>RA</sub> & t <sub>WA</sub>	0		t <sub>CA</sub>	60
	t <sub>DW</sub>	200		t <sub>DW</sub>	30
	t <sub>WD</sub>	0		t <sub>WD</sub>	60
	t <sub>RR</sub>	250		t <sub>CC</sub>	*180
	t <sub>WW</sub>	200		t <sub>CC</sub>	30
	t <sub>AR</sub>	0		t <sub>AC</sub>	115
	t <sub>AW</sub>	20		t <sub>AC</sub>	95
8271 & 8273	t <sub>AD</sub>		200	t <sub>AD</sub>	*350
	t <sub>RD</sub>		150	t <sub>RD</sub>	*200
	t <sub>CA</sub>	0		t <sub>CA</sub>	60
	t <sub>DW</sub>	150		t <sub>DW</sub>	80
	t <sub>WD</sub>	0		t <sub>WD</sub>	80
	t <sub>RR</sub> & t <sub>WW</sub>	250		t <sub>CC</sub>	*180
	t <sub>AC</sub>	0		t <sub>AC</sub>	115
8275	t <sub>RD</sub>		200	t <sub>RD</sub>	*150
	t <sub>RA</sub> & t <sub>WA</sub>	0		t <sub>CA</sub>	60
	t <sub>DW</sub>	150		t <sub>DW</sub>	80
	t <sub>WD</sub>	0		t <sub>WD</sub>	60
	t <sub>RR</sub> & t <sub>WW</sub>	250		t <sub>CC</sub>	*180
	t <sub>AP</sub> & t <sub>AW</sub>	0		t <sub>AC</sub>	115
8279-5	t <sub>AD</sub>		250	t <sub>AD</sub>	300
	t <sub>RD</sub>		150	t <sub>RD</sub>	200
	t <sub>RA</sub> & t <sub>WA</sub>	0		t <sub>CA</sub>	60
	t <sub>DW</sub>	150		t <sub>DW</sub>	80
	t <sub>WD</sub>	0		t <sub>WD</sub>	60
	t <sub>RR</sub> & t <sub>WW</sub>	250		t <sub>CC</sub>	*180
	t <sub>RCY</sub>	1000		t <sub>RV</sub>	**
	t <sub>AP</sub> & t <sub>AW</sub>	0		t <sub>AC</sub>	115

Table 6. Peripherals vs. 8085A-2

\*With 1 "Wait State"

\*\*Must allow for in Software

Taking note of asterisked margins shown on the comparison sheet:  $t_{AD}$ ,  $t_{RD}$ ,  $t_{RR}$  and  $t_{DW}$ , it is seen that they are all taken care of by introducing a wait state. The double asterisked margins deal with the  $t_{RV}$  spec on the 8255A-5, 8253-5 and 8279-5 peripherals.  $t_{RV}$  is the time from the rising edge of  $WR$  or  $RD$  to the next falling edge. To allow sufficient time for this spec it is necessary to delay the commands sent to these three peripherals. Enough dead time must occur to make up for the entire negative portion of the margin (for example: 790ns in the 8253-5 medium system). Since in the 8085A-2 every machine cycle is at least 200ns long, 4 machine cycles are sufficient time to allow peripheral control signal recovery ( $t_{RV}$ ).

One may notice that all of the 8085A instructions take at least 4 T-states (providing a minimum of 800ns) giving ample time to meet this requirement, just by programming one instruction in between every command sent to the peripheral. I/O mapped I/O, which results in using the Input, Output instructions has this delay time built in when moving the data to be transferred into the accumulator. With memory mapped I/O, any instruction that accesses memory for data will provide the time necessary to not violate  $t_{RV}$  as a second fetch is performed.

#### Bus - Loading Considerations - Decoupling

For the cost conscious designer it is always helpful to know when buffering is needed and when it is not. How much can I load the 8085A output pins down? To answer this it is helpful to first list the DC requirements of the common types of logic loading and compare this to the capabilities of the 8085A.

	Maximum High-Level Input Current	Maximum Low-Level Input Current
TTL (single load)	40 $\mu$ A	1.6mA
Schottky or HTTL	40 $\mu$ A	2.0mA
MOS	10 $\mu$ A	10 $\mu$ A
LSTTL (single load)	20 $\mu$ A	400 $\mu$ A

The 8085A is capable of an IOL of 2mA (low) and IOH of - 400 $\mu$ A. With this spec it is easy to come up with the possible combinations of D.C. loading that the designer can use without buffering:

LOADS	8085A, A-2 limiting factor (level)
1 TTL + 1 LSTTL	LOW
1 TTL + 36 MOS*	HIGH
1 SCHOTTKY or 1 HTTL	LOW
40 MOS (various combinations possible)*	HIGH
5 LS TTL	LOW

\* Exceeds capacitive loading limit, to be discussed

If a user exceeds these DC loading limitations he must buffer that particular signal. Another factor that the designer **must** consider is the capacitive load that is seen by the 8085A outputs, which may very well be excessive even if DC loading is not. One may note that even though the 8085A can handle a DC load of 40 MOS devices or 36 MOS + 1 TTL, their collective input capacitances exceed the 150 pF max spec.

The timing specs of the 8085A are guaranteed as long as the 150 pF maximum loading is not exceeded, which includes the wires, components and parasitics. If the user exceeds this value and wants to guarantee his system timing he must either derate the system timings or use buffering.

What if you choose to ignore this limit and say you can live with the performance degradation? First the timing performance is not all that would degrade, a user must be willing to give up some reliability of his components (All MOS devices have this restraint). This is caused by the excessive switching currents that are needed for this extra loading capacitance. If reliability is not an important consideration, the user can load up to 300 pF on the 8085A bus, but the following correction factors must be used to adjust the timings:

for 150 pF < 300 pF add .13 ns/pF

conversely if less than 150 pF:

for 25 < CL < 150 pF you can subtract .1/ns/pF.

What happens after 300 pF? If the user exceeds this, the noise levels become excessive and problems will result. How much is too much noise? 350 mvolts zero to peak. Prudent designers will always buffer when noise approaches this level, especially in the case of going from one board to another.

The above takes into consideration the actual specification considerations of when to buffer, but there are also transmission line and noise effects that must be considered. When working with dynamic RAMs small (20-30 ohm) resistors are commonly put in series in the address lines to help match impedance levels and reduce reflections. Note that this resistor should be chosen such that it does not severely degrade the voltage levels of the signal. Long parallel board traces with signals that could adversely affect each other should also be avoided to prevent cross talk problems.

By-passing is very important to prevent intermittent problems which often plague the board designer. Large bulk capacitors should be used at strategic locations on the board to prevent power supply droop. This becomes a major factor when there are many devices that can turn on at once and produce a considerable drain from the power supply (such as burst refresh in dynamic RAM).

To help smooth out the current spikes that naturally occur when devices turn on and off, it is recommended to liberally use small capacitors such as the monolithic and other ceramic capacitors which have low inherent inductance. Attached in the 2117 data sheet is a suggested layout of capacitors to effectively bypass the supply lines to ensure proper system operation. Cutting corners here will often times turn around and bite you.

Proper layout is an important consideration. Power supply lines should be well gridded to supply sufficient current to all areas of the board. A strong ground layout is advised to offset noise problems. Remember if the ground plane moves up in voltage because of excessive charge dumping in a particular area, the supply will drift up correspondingly. Sensing low levels often becomes an intermittent problem when proper ground is not provided.

## Overview

Following is an application example that illustrates the use of the interrupt and SOD pins on the 8085A, software for a block search routine, and the procedure for using and reading the 8155 counter. It is a simple application showing the use of the small but powerful 3-chip MCS-85 system as a temperature sensor (SDK-85 board used). This example can be modified to be an accurate industrial temperature controller, for several locations if desired.

The basic operation behind this application is a monostable multivibrator having its timing pulse duration controlled by a thermistor. The counter in the 8155 converts this timing pulse to a decimal count that is software mapped into a temperature and displayed in degrees C in the address field of the display in the SDK-85 Kit. For the purpose of keeping the software relatively simple, many approximations were incorporated into the code.

## Detailed Hardware

The basic SDK kit was used for the initial hardware. This Kit provides for everything necessary to develop and debug a program through the use of the SDK-85 monitor, keyboard and display board. The kit provides for 256 bytes of RAM resident in the 8155 and 2K bytes of ROM or EPROM where the SDK-85 monitor is placed. (See the Intel SDK-85 User's Manual for copy of monitor software code.)

Figure 20 is a schematic of the SDK-85 Kit with only one 8155 and 8355. There is no buffering in this system as all compo-

nents are on the same board and far below the maximum component loading. A monostable multivibrator (74121) is also shown with a thermistor connected to RE/CE.

The SOD output pin from the 8085A is used for the purpose of starting the monostable multivibrator in generating its temperature controlled timing pulse. This pulse is created by the RC time constant provided for by the thermistor acting as a variable resistor and a .1 $\mu$ F capacitor to put the timing pulse in the desired timing range.

The inverted output of the monostable multivibrator (one shot) has been directly connected to the RST 6.5 pin on the 8085A. Since this pin is high level sensitive, it is necessary to disable interrupts in the program until after the pulse from the one shot goes low.

The hardware addressing in the configuration shown allows for several code spaces that could be used. The RST and TRAP interrupt lines on the 8085A also have hardware start addresses but many of these are altered by the SDK monitor. Table 7 should be useful in understanding the addresses used in the software that follows. Each memory/ I/O component in the basic SDK-85 system is enabled by a signal coming from the 8205 address decoder. Since no expansion chips are used, output enables 00 (8355 monitor ROM), 03 (8279 Keyboard) and 04 (8155 RAM) were the only ones needed. Additional memory and/or I/O could have been incorporated using other output enables from the 8205.

Memory/ I/O Device	Function	Output from 8205	code space
8155	RAM space	04	2000 - 20FF (20 - 20FF are reserved for monitor RAM locations)
8355	ROM space	00	0000 - 07FF
8279	Keyboard/display controller	03	1800 - 1FFF
stack pointer			
Since the monitor uses locations 10C8 through 20FF, the stack pointer must be initialized to 20C8 or less.			
8085A jump address	Usage	monitor mapped address	
trap	T0 of 8155	0157	
RST 5.5	2CH 8279 interrupt	028E	
RST 6.5	34H oneshot interrupt	20CE	
RST 7.5	3CH vector interrupt		
I/O ports address	Function		
00	Monitor ROM Port A (8355)		
01	Monitor ROM Port B (8355)		
02	Monitor ROM Port A (8355) Data direction register		
03	Monitor ROM Port B (8355) Data direction register		
20	Basic command/status register		
21	Basic RAM Port A		
22	Basic RAM Port B		
23	Basic RAM Port C		
24	Basic RAM LOW order byte of timer count		
25	Basic RAM HIGH order byte of timer count		

Table 7. Addressing

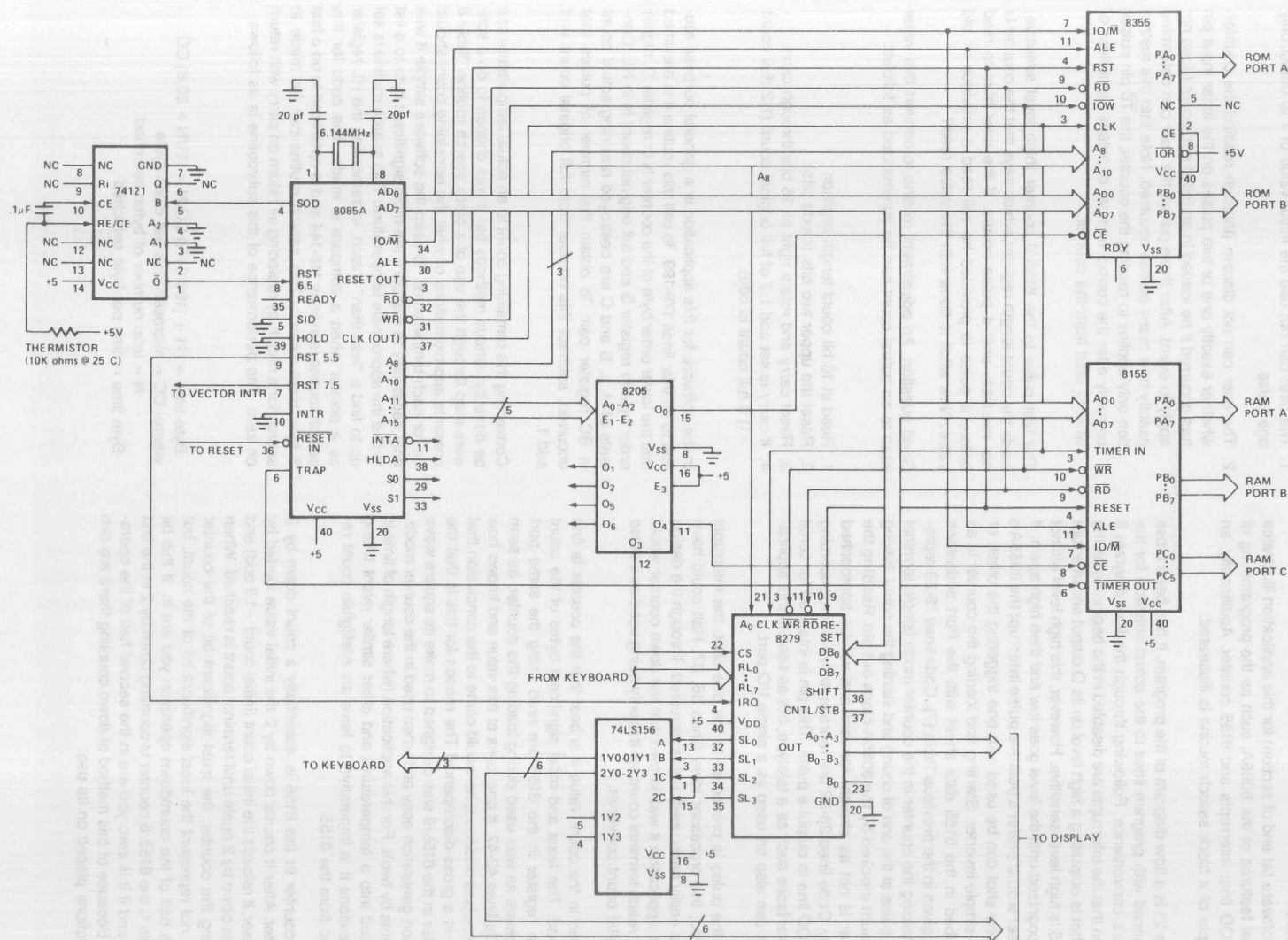


Figure 20. Detailed SDK-85 Kit with Temperature Sensor



## Software

The software (at end of section) for this application illustrates several features of the 8085A, such as the programming of the SOD line, interrupts and 8155 counter. Additionally, an example of a block search routine is illustrated.

Figure 21 is a flow diagram of the program. It has been cross referenced with program lines to the actual software for the reader's convenience. Following through the flow diagram it is seen that the interrupts are disabled in the beginning as the one shot is outputting a high level on its Q output and interrupt pin 6.5 is high level sensitive. However, this high level will not be recognized until the level goes low and then high again. If the user would prefer a positive pulse interrupt the 8085A a dual one shot can be used with one triggering the other, or just a simple inverter. Starting and loading the counter is as described in the 8155 data sheet with the Port addresses being given in the previous Table (7). Code lines 18-23 represent placing the counter in the counter mode (single terminal count pulse at the end of count) and starting the count, having the count clocked by the 8085A clock out pin. Reading the counter is not as straight forward and will be approached shortly. Code lines 28-32 are representative of programming the SOD line to output a pulse. This pin is intended for serial I/O interfaces such as a teletype, but as seen in this application, it can also be used as a single I/O port.

After the pulse is presented to the one shot, the interrupts enabled, the processor idles (lines 36, 37; Halt could have just as easily been used) until interrupted. Through the design of this application it was known that the down counter would never reach terminal count, as it is only being used as a pulse to digital count converter.

To read in the count value it is best that the counter is first stopped. The least and most significant bytes of the count length register in the 8155 are read using the same port addresses as was used during loading the counter, as seen in code lines 42-47. If one looks at this value and knows how many pulses occurred, he would come to the conclusion that there is a gross discrepancy! The reason for this is that the counter in the 8155/6 was designed to make its square wave function generation easy and when used in the counter mode, it counts by two's. For this application (where length of time is mapped into a temperature) and other similar event timing applications it is imperative to have an intelligible count returned from the 8155.

The counter in the 8155 is essentially a count down by 2 counter. After it counts down by 2 the initial value loaded by the user, it reloads the initial count (initial count - 1 if odd) and counts down by 2 again until terminal count is reached. When reading the counter, the least significant bit of the counter does not represent the least significant bit of the count, but which half of the countdown operation you are in. If this bit equals 1, the 8155/6 counter is counting down by 2 in the first half, and if it is zero you are in the second half of the operation. Because of this method of down counting there are two restrictions placed on its use:

1. The user can not use the initial value of 1 to detect only one pulse.
2. The user can not discern (through reading the counter) whether exactly one or two pulses on the timer input pin has occurred if he loaded in an initial odd count (does not apply to even). After three pulses the user can determine exactly how many pulses occurred. Note that this restriction only applies to reading the counter, the T0 pin pulses correctly after the correct number of pulses regardless of what is read from the counter.

The first pulse to the 8155/6 counter (high level sensitive) loads the count length register, which says that the counter is not readable until a pulse occurs. If the user tries to read before a pulse is provided he will read a previous or old value. Now what is done with the value read?

Good question. An adjustment routine to convert this value read to an actual count can be summarized as follows:

1. Read in 16 bit count length register.
2. Reset the upper two bits (mode bits).
3. Reset carry and rotate right all 16 bits through carry.
4. If carry is set add 1/2 of full original count (1/2 (full count - 1) if full count is odd).

In the software for this application is a general purpose routine to do this; lines 179-199. To call this routine it is assumed that the lower order byte of the counter is in register C, higher order byte in register B and full original count is in HL. Contents of H, L, B and C are destroyed returning actual count in BC register pair. To obtain the number of pulses that occurred, subtract this number from full original count and add 1.

Converting this remaining count to an actual temperature can be done by various methods but it was chosen to do a software map through the use of a block search routine. Table 8 presents approximations of what the remaining count should be for each temperature. To keep the software simple it was only necessary to compare the most significant byte to a list to find the appropriate temperature. This search routine is set up to find a "less than" match, incrementing the HL register as a pointer when a compare is made. The code for this search routine is in lines 118-144 and is optimized to be a fast 8 byte block search. This search routine can be made to search for a match by replacing all return on carry with return on zero. The performance of this subroutine is as follows:

$$\text{Byte time} = (11 + (166/8) N) \text{ CC}/N = (11/N + 20.8) \text{ CC}$$

where: CC = microseconds per clock cycle

N = total number of bytes searched

Byte time = time per byte searched

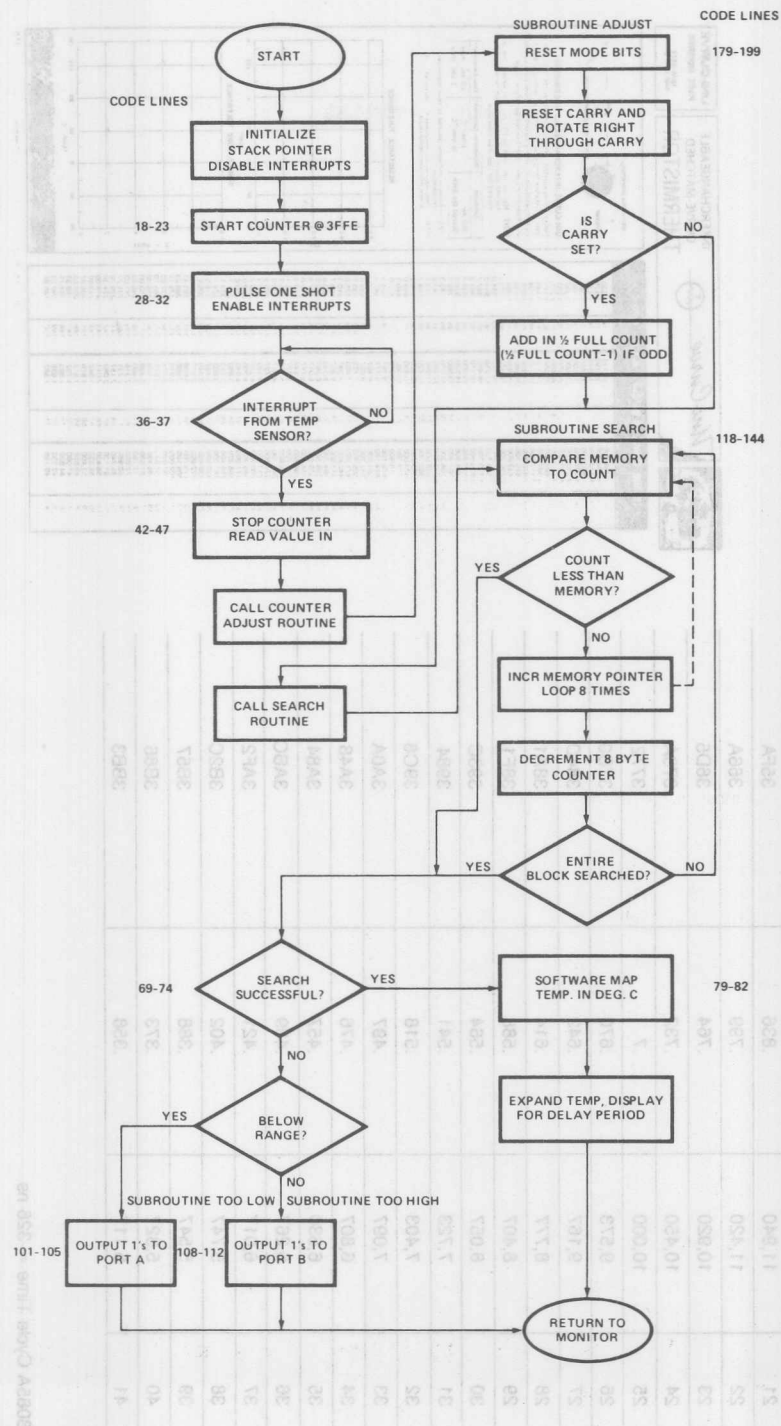


Figure 21. Temperature Sensor Flow Diagram

DEG. C	THERMISTOR OHMS	(.7) (.1μ) (R <sub>T</sub> ) APPROX. TIME (ms)	START WITH 3FFE <sub>H</sub> APPROX. COUNT LEFT (HEX)
20	12,490	.874	3585
21	11,940	.836	35FA
22	11,420	.799	366A
23	10,920	.764	36D5
24	10,450	.732	373A
25	10,000	.7	3772
26	9,573	.670	37D0
27	9,167	.642	384D
28	8,777	.614	38A1
29	8,407	.588	38F1
30	8,057	.564	393C
31	7,723	.541	3984
32	7,403	.518	39C8
33	7,097	.497	3A0A
34	6,807	.476	3A48
35	6,530	.457	3A84
36	6,267	.439	3ABC
37	6,017	.421	3AF2
38	5,747	.402	3B2C
39	5,547	.388	3B57
40	5,327	.373	3B86
41	5,117	.358	3BB3

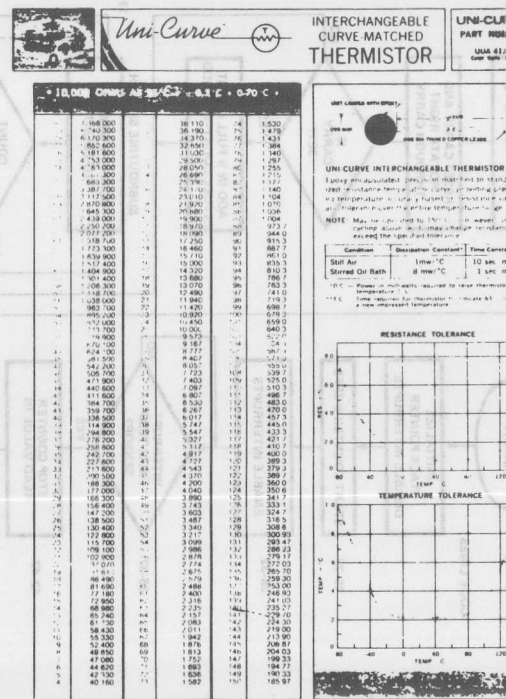
8085A Cycle Time = 326 ns

Oneshot Approx. Time =

L<sub>N2</sub> (CEXT) (REXT)

≈ (.7) (.1μ) R<sub>THERMISTOR</sub>

Table 8. Thermistor Resistance Mapping



Search	Cmp M	compare byte
	RZ	return if match
	INX H	else increment pointer
	DCR C	has the entire
	JNZ search	block been searched?
	STC	If so set no match flag
	RET	and return.

BLKMV	LXI H, 000H	clear HL
	DAD SP	move SP to HL
	SHLD SAVESP	save sP
	MOV H, B	move Block move
	MOV L, C	Source address
	SPHL	To SP
	XCHG	Move Block move
		address to HL

Loop	POP B	fetch four bytes from
	POP D	source store 1st byte
	MOV M, C	at destination
	INX H	
	MOV M, B	2nd
	INX H	
	MOV M, E	3rd
	INX H	
	MOV M, D	4th
	INX H	
	DCR A	check for end of
	JNZ Loop	Block move
	LHLD SAVE\$P	return old
	SPHL	SP
	RET	return

## APPLICATION EXAMPLE 2

### CRT INTERFACE

Most microprocessor systems require some sort of serial communications. This may be selected for reasons of economy (to reduce the number of interconnections required in a distributed system), or it may be necessary in order to communicate with such common peripherals as CRT's or teletypewriters.

These peripherals all use a standard convention for transmitting serial ASCII code. Each data byte is transmitted as a series of 10 or 11 bits. The uniform time per bit corresponds to the data transmission rate. For example, if the transmission rate is to be 2400 baud (2400 bits per second), each bit time must be  $1/2400 \text{ bps} = 416.7 \mu\text{sec/bit}$ . The standard 10-bit sequence consists of a logically zero "Start" bit, 8 data bits (least significant bit first), and one or more stop bits (logic 1). An 11-bit sequence with two stop bits is used for 110 baud TTY's. The logic one level continues until the start bit of the next byte to ensure that each 10-bit sequence is initiated with a one-to-zero transition. The 8 bits transferred might be raw binary data or alphanumeric characters using the standard ASCII code. In this case, the most significant bit — the last data bit transmitted — will depend on the parity convention being used. This sequence is illustrated for the ASCII "space" character in Figure 22.

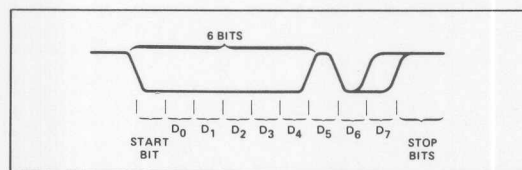


Figure 22. ASCII Space Character

The algorithm for receiving serial code involves sampling the incoming data at the middle of each bit time. The eight sampled values are shifted into a serial byte corresponding to the data originally transmitted. The one-to-zero transition at the beginning of each byte makes it possible to synchronize the sampling points relative to the start of each data sequence.

### Hardware Interface

In general, any serial communications system will require both hardware and software interfaces. Since the SOD line can drive only one TTL load, additional current and voltage buffering is required to be compatible with the RS-232C interface standard used by most peripherals. A schematic for achieving this buffering is shown in Figure 23. The MC1488 and MC1489 circuits interface positive logic TTL signals with the RS-232 high voltage inverted logic levels.

### Software Package

The software needed to drive the CRT interface is divided into three parts. All three use software timing and delay loops, with fixed and variable parameters. In conjunction, they are able to identify incoming signals at any rate from below 110 to over 9600 baud and respond at the same rate.



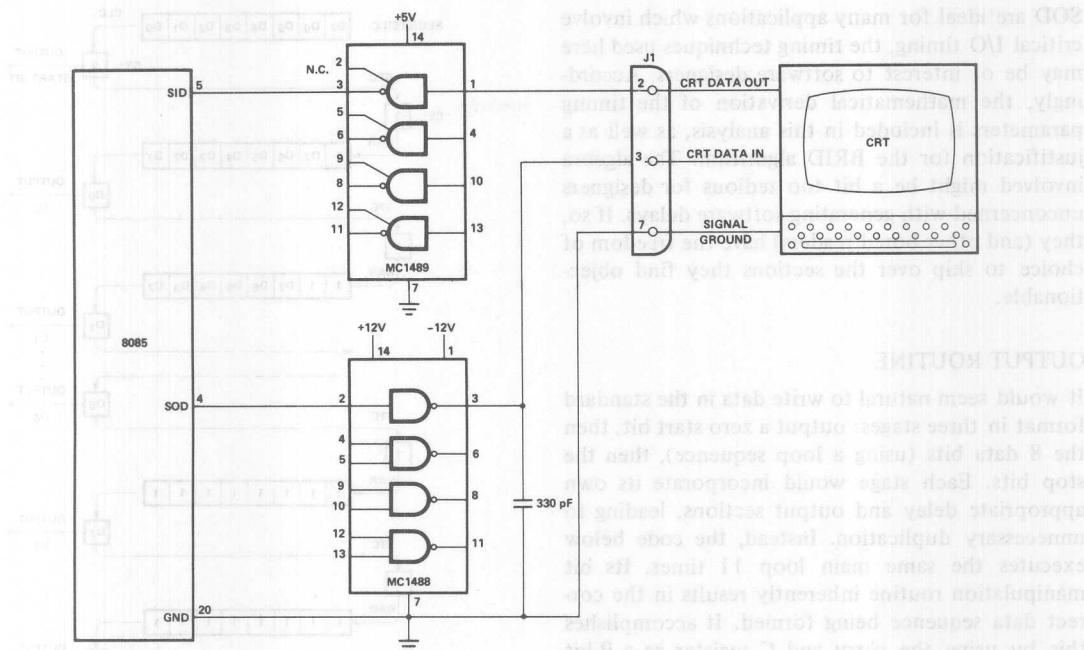


Figure 23. RS-232C Interface Schematic

Upon power-up or reset, or when the console device baud rate is changed, the baud rate identification subroutine (BRID) is called. This routine waits until an ASCII space character (20H) is received from the console. (Any other character will result in a case of mistaken identification.) When a space character is received, two time parameters are computed which correspond to the bit time and one-half the bit time of the baud rate being used. These are stored as variables BITTIME and HALFBIT. To output a character to the console, the character code is placed in register C, and the subroutine COUT is called. This routine uses BITTIME as a parameter for the software delay loop which determines the baud rate. To accept a character from the keyboard, CIN is called. CIN returns after the next key is typed, with the corresponding character code in register C. CIN uses both parameters BITTIME and HALFBIT.

Since COUT and CIN use time parameters computed by BRID, they will function at a rate the same as that of the initial space character input. Because of the nature of the software, the rate does not depend on the CPU clock frequency. This

results in additional flexibility in the following respects:

1. The software does not need to be modified if the 8085 crystal frequency is changed or Wait states are added.
2. Since the time base is no longer critical, the quartz crystal could be replaced by a less expensive RC network, provided the frequency does not drift by more than a few percent during a session. Additional drift can be accommodated by periodically recalling the BRID routine.
3. Communication is possible at non-standard baud rates which relaxes the constraints on system peripherals.

It should be noted, though, that slowing down the CPU clock will decrease its throughput proportionately. In addition, it will degrade the maximum resolution of the delay loops, with the result that the highest baud rates may no longer be achievable.

A more detailed analysis of the CRT interface routines will be presented in the order of increasing complexity: COUT, CIN, and BRID. Since SID and

may be of interest to software designers. Accordingly, the mathematical derivation of the timing parameters is included in this analysis, as well as a justification for the BRID algorithm. The algebra involved might be a bit too tedious for designers unconcerned with generating software delays. If so, they (and other bored readers) have the freedom of choice to skip over the sections they find objectionable.

## OUTPUT ROUTINE

It would seem natural to write data in the standard format in three stages: output a zero start bit, then the 8 data bits (using a loop sequence), then the stop bits. Each stage would incorporate its own appropriate delay and output sections, leading to unnecessary duplication. Instead, the code below executes the same main loop 11 times. Its bit manipulation routine inherently results in the correct data sequence being formed. It accomplishes this by using the carry and C register as a 9-bit pseudo-circular shift register. Initially CY=0. The algorithm outputs CY, waits one bit time, sets CY=1, and then rotates the pseudo-register right one bit. This repeats for 11 cycles. On the tenth and all subsequent loops, the output bit will be a logical one, since that bit had been set nine loops earlier while in the CY (see Figure 24).

When COUT is called the registers to be used must be preserved and interrupts disabled so the timing loop will not be disrupted. Clear the CY in preparation for outputting the start bit, and set the loop counter for 11 bits (if 110 baud will never be used, the counter could be set to 10):

```
COUT:  PUSH    B
        PUSH    H
        DI
        MVA    A, B
        MVI    B, 11
```

Output of the contents of the CY:

```
001:  MVI    A, 00H (7)
        RAR    (4)
        SIM    (4)
```

The numbers in brackets indicate how many machine cycles are required for each instruction. They will be referred to in the timing analysis section.

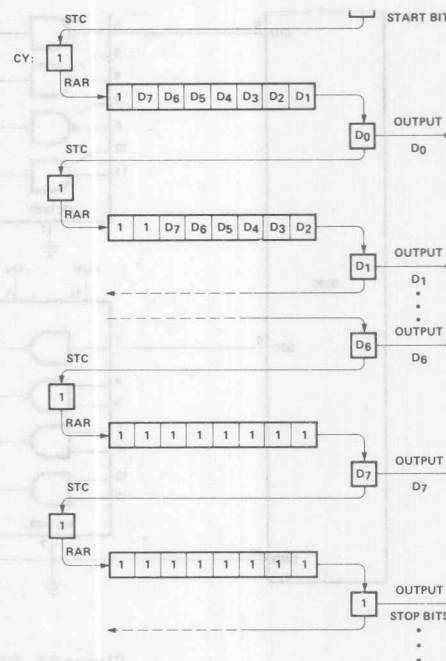


Figure 24. Data Serialization Algorithm

Get stuck in a loop for the appropriate time (don't worry for now how "BITTIME" is determined):

```
002:  LALO    BITTIME (16)
        DCR    L (4)
        JNZ    002 (4)
        DCR    H (4)
        JNZ    002 (4)
```

Rotate the contents of register C right into the CY, while moving a one into the left end. Continue until all bits have been transmitted:

```
STC (4)
MOV    A, C (4)
RAR    (4)
MOV    C, A (4)
DCR    B (4)
JNZ    001 (10)
```

Restore processor status and return:

desired. This guarantees that the baud rate is accurate to within 0.3%.  
 POP R; EI; RET

## INPUT ROUTINE

The console input routine uses the opposite procedure; instead of moving a bit from register C to the CY, then to A7, then to SOD, CIN loads a bit from SID into A7, then moves it to CY, then into register C.

First, set up the CPU as before:

```
CIN: PUSH H
      DI
      MVI B, 9
```

When a start bit transition arrives, the first sampling should not be taken until the middle of the first data bit, one and one-half bit times after the transition. Await the start bit transition, then set up the delay parameter for one-half bit time:

```
(7) C11: RIM          (4)
      ORA A         (4)
      JM C11        (7)
      LHL HALFBIT  (16)
```

Loop for one-half bit time before starting to sample data:

```
C12: DCR L          (0)
      JNZ C12       (0)
      DCR H          (0)
      JNZ C12       (0)
```

Wait until the middle of the next bit before sampling SID, then move the data bit into CY:

```
C13: LHL BITTIME  (16)
C14: DCR L        (0)
      JNZ C14     (0)
      DCR H        (0)
      JNZ C14     (0)
      RIM          (4)
      PHL          (4)
```

Decrement the bit counter. If this is the ninth cycle, the 8 data bits are in register C, so quit (the first stop bit will already have been received, and be in CY):

```
      DCR B        (4)
      JZ C15       (7)
```

Otherwise, continue. Rotate the data bit right into register C, and repeat the cycle:

```
EXAMPLE:
MOV A, C    (4)
RAR         (4)
MOV C, A    (4)
NOP         (4)
JMP C13     (10)
```

```
C15: POP H      (4)
      EI        (1)
      RET
```

(A NOP is needed to make the COUT and CIN loops exactly equal in number of machine cycles, so that each can use the same delay parameter.) Restore status and return.

## TIMING ANALYSIS

COUT and CIN now need to be provided with parameters for BITTIME and HALFBIT. It can be seen from the above code that each routine uses  $61 + D$  machine cycles per input or output bit, where  $D$  is the number of cycles spent in either four line delay segment. If  $\langle H \rangle$  and  $\langle L \rangle$  are the contents of the H and L registers going into this section of code, then:

$$D = 22 + (\langle L \rangle - 1) \times 14 + (\langle H \rangle - 1) \times [(255 \times 14) + 25] \quad (1)$$

If  $\langle H' \rangle \equiv \langle H \rangle - 1$ ,  $\langle L' \rangle \equiv \langle L \rangle - 1$ , and

$$\langle HL' \rangle \equiv 256 \langle H' \rangle + \langle L' \rangle \quad (2)$$

then

$$D = 22 + 14 \langle L' \rangle + 3595 \langle H' \rangle \quad (3)$$

This can be approximated by:

$$D = 22 + 14 \langle HL' \rangle \quad (4)$$

This approximation is exact for  $\langle H' \rangle = 0$ ; otherwise, it is accurate to within 0.3%. Thus each loop of COUT or CIN uses a total of:

$$C = 61 + D = 83 + 14 \langle HL' \rangle \text{ machine cycles} \quad (5)$$

Each machine cycle uses two crystal cycles in the 8085, so the resulting data rate is:

$$B = \frac{\text{cycle frequency}}{C} = \frac{(\text{crystal frequency}) \div 2}{83 + 14 \langle HL' \rangle} \quad (6)$$

For a typical calculation, see the example below.

#### EXAMPLE

To produce 2400 baud with the standard 6.144 MHz crystal:

$$2400 = \frac{(6.144 \times 10^6) \div 2}{83 + 14 \langle HL \rangle'}$$

$$14 \langle HL \rangle' = \left( \frac{6.144 \times 10^6 \div 2}{2400} \right) - 83$$

$$\langle HL \rangle' = \left[ \left( \frac{6.144 \times 10^6 \div 2}{2400} \right) - 83 \right] \div 14 = 85.5 \cong 86$$

$$\langle HL \rangle' = 86_{10} = 0056H$$

$$\langle HL \rangle = 0157H = \text{BITTIME}$$

To determine the true data rate this parameter will produce, substitute into equation (6):

$$\begin{aligned} \text{Data Rate} &= \frac{6.144 \times 10^6 \div 2}{83 + 14(86)} \\ &= 2387 \text{ baud, which is } 0.54\% \text{ slow.} \end{aligned}$$

For 9600 baud, the same calculations will yield  $\langle HL \rangle' = 17$ , which is actually 0.3% slow; a sizzling 19200 baud or 38400 baud could each be generated to within 5% if  $\langle HL \rangle' = 6$  or 0! Table 9 presents the parameters for several standard baud rates.

Notice that the resolution of the delay algorithm — the difference between bit times resulting from parameters which differ by one — is 14 machine cycles. As a result, the true bit delay produced can always manage to be within  $\pm 2.3 \mu\text{sec}$  of the delay

desired. This guarantees that at rates up to 9600 baud, where each bit time is at least  $104 \mu\text{sec}$  wide, some value of BITTIME can be found which will be accurate to within 2.2%.

#### BAUD RATE IDENTIFICATION ROUTINE

The function of BRID is to compute the appropriate parameters BITTIME and HALFBIT. It accomplishes this by observing the data pattern received when the space bar is pressed on the console device. Since a space character has the ASCII code  $20H = 00100000B$ , the pattern represented back in Figure 4 is transmitted. Notice that the initial zero level is 6 bits wide. Suppose it could be determined that this corresponds to  $M$  machine cycles. Then one bit would correspond to  $(M \div 6)$  machine cycles. The reason for dividing down a space several bits long is so that any distortion caused by the signal rise and fall times, or any lack of precision in detecting the two transitions, will be reduced by a factor of six. Since the bit period of COUT and CIN is  $83 + 14 \langle HL \rangle'$ , BRID must generate a value  $\langle HL \rangle'$  such that:

$$M \div 6 = 83 + 14 \langle HL \rangle' \quad (7)$$

$$\langle HL \rangle' = \frac{(M \div 6) - 83}{14} \quad (8)$$

$$\langle HL \rangle' = \frac{M}{84} - 6 \text{ (approximately)} \quad (9)$$

This value can be determined by setting register pair HL to  $-6$ , then incrementing it once every 84 machine cycles during the period that the incom-

Table 9

#### DELAY PARAMETERS FOR STANDARD BAND RATES USING 6.144 MHz CRYSTAL

TARGET BAUD RATE	$\langle HL \rangle'_{10}$ (See Text)	$\langle HL \rangle'_{16}$ (See Text)	$\langle HL \rangle'$ or BITTIME (See Text)	HALFBIT	ACTUAL BAUD RATE PRODUCED	% ERROR
110	1989	07C5	08C6	04E3	109.99	-0.006
150	1457	05B1	06B2	03D9	149.99	-0.005
300	726	02D6	03D7	026C	299.80	-0.068
600	360	0168	0269	01A5	599.65	-0.059
1200	177	00B1	01B2	0159	1199.5	-0.039
2400	86	0056	0157	012C	2386.9	-0.547
4800	40	0028	0129	0115	4777.6	-0.469
9600	17	0011	0112	0109	9570.1	-0.312
19200	6	0006	0107	0104	18395.2	-4.37

ing signal is zero. BITTIME is then obtained by individually incrementing registers H and L. To obtain HALFBIT, divide the value of <HL> determined above by two before incrementing each register.

In order to implement this algorithm, set HL to -6, verify that the incoming signal is a logic one, then wait for the start bit transition.

```
BRID: MVI A, 0C0H
      SIM
      LYI H, -6H
BR11: RIM
      ORA A
      JP BR11
BR12: RIM
      ORA A
      JM BR12
```

Increment register pair HL, then delay so that each cycle will require 84 machine cycles:

```
BR13: INX H
      MVI E, 04H
BR14: DCR E
      JNZ BR14
```

Check if SID is still low. If so, repeat:

```
VOLUME CONTROL
      RIM
      ORA A
      JP BR13
```

Otherwise continue. Store HL temporarily for the HALFBIT calculation. Obtain and store BITTIME:

```
PUSH H
INR H
INR L
SHLD BITTIME
```

Restore HL, calculate HALFBIT, and return:

```
POP H
ORA A
MOV A, H
RAR
MOV H, A
MOV A, L
RAR
MOV L, A
INR H
INR L
SHLD HALFBIT
RET
```

The assembled listings for these subroutines, along with a simple test program, is presented in the CRT and Cassette Code.



transmitted through a non-ideal medium. To give three typical examples, a system with electrically isolated elements might require that signals be AC coupled, communications through an audio network (such as telephone or radio) are greatly bandwidth limited, and some applications (such as a distributed network in an industrial environment) must tolerate random electrical noise. Attempting to record data on a cheap cassette recorder (the one used for this note cost \$17.00) will reveal all of these shortcomings, plus one: The tape speed fluctuates significantly and varies as the batteries run down, hence the data rate is inconsistent.

The recording scheme used here makes very few demands on the transmission medium. It makes no attempt to transmit DC voltage levels. Instead, data is transmitted by a series of variable length tone bursts. The dominant frequency of the tone used can be selected to be within the passband of the particular medium. Data is transmitted with each bit composed of a tone burst followed by a pause. The first third of a bit period is always a tone burst, the middle third is either a tone burst continuous with the first or a pause corresponding to, respectively, a one or zero, and the final third is always a pause, as shown in Figure 25. Thus, data is distinguished by the burst/pause ratio.

### Hardware Design

These tone bursts are obtained from the 8085 SOD line, using analog signal conditioning to eliminate the DC component of the waveform. (This low frequency component is due to the single-ended nature of the SOD line: its deviations from ground are all positive, which unbalances the capacitive input stage of the recorder.) A suggested interface

quad op amp and a few standard value discrete components which should be available in even a digital design laboratory. On playback, analog circuitry is again used to detect the presence of a tone burst. In Figure 26, A2 buffers the incoming signal, and A3 inverts it. The peaks of these two signals are transmitted through D1 or D2 and are filtered by an RC network. Comparator A4 then squares up the output and produces the logic signal read by the SID pin. Since the op amps are powered by the single 5-volt supply, a 2.0-volt reference level is obtained from a resistive voltage divider. The waveforms present at several points in the circuit are shown in Figure 27.

### Software

The algorithm for reading a data bit off the tape is simple and straightforward: If the tone burst is longer than the pause, the bit is a one. Otherwise, it is a zero. Since only the time ratio is considered, any variation in tape speed will not affect the data determination.

### VOLUME CONTROL

A question that arises with any audio cassette interface is how to set the volume control. (Recording level is usually determined internally.) When the playback level is correct, the logic signal output from A4 will have either a one-third or two-thirds duty cycle. This can be readily observed with an oscilloscope. In the field, an old-fashioned mechanical-type voltmeter could be connected to the A4 output, and the volume adjusted until the meter needle hovered somewhere between 1/3 and 2/3 the high level output voltage. With random data, the reading would be about 2 volts. There will be a fairly wide range of acceptable volume settings. (Since the quivering meter needle is being used here for inertial signal averaging, a digital voltmeter would not be very helpful in this application.)

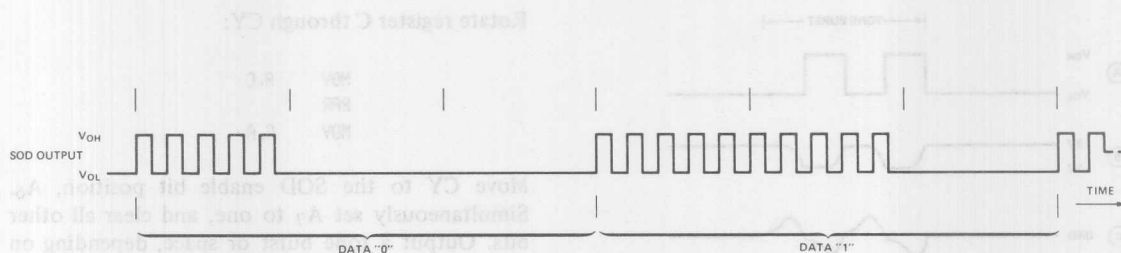


Figure 25. Tape Interface Data Recording Scheme

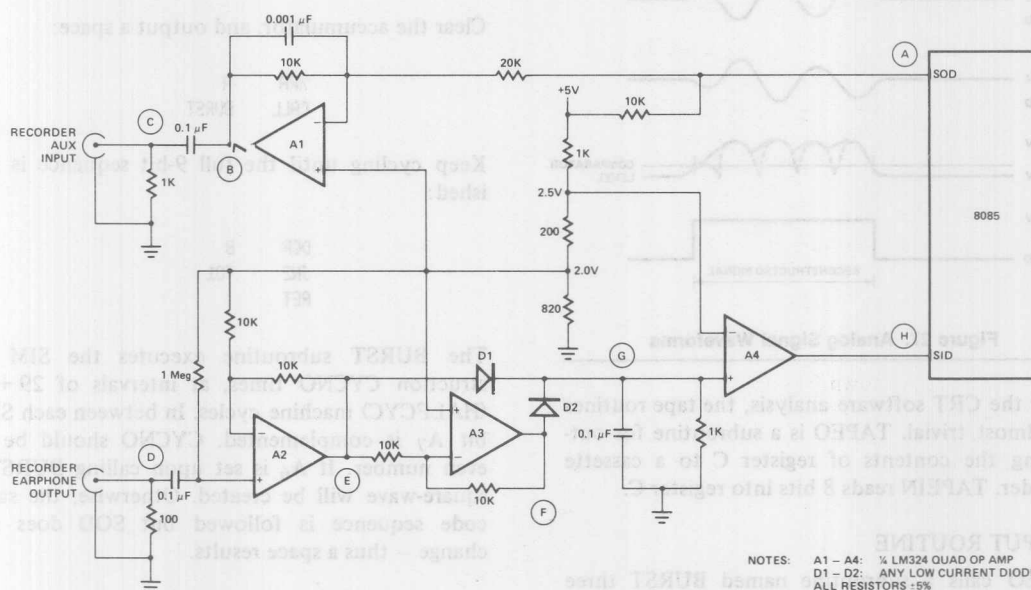


Figure 26. One Chip Magnetic Tape Interface Schematic

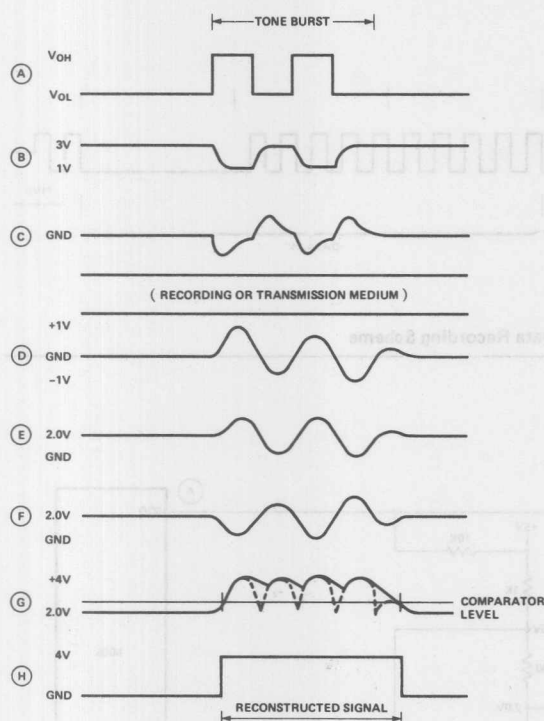


Figure 27. Analog Signal Waveforms

After the CRT software analysis, the tape routines are almost trivial. TAPEO is a subroutine for outputting the contents of register C to a cassette recorder. TAPEIN reads 8 bits into register C.

#### OUTPUT ROUTINE

TAPEO calls a subroutine named BURST three times for each bit. If A<sub>6</sub> (the SOD enable bit) is set when BURST is called, a square-wave tone burst will be transmitted. If A<sub>6</sub> is not set, BURST simply delays for exactly the same amount of time before returning. The three calls are used to, respectively, output the initial burst, output the data burst/space, and create the space at the end of each bit. Nine bits will be output: the eight data bits (LSB first) followed by a zero bit. The start of the initial burst of the trailing zero is needed to mark the end of the final space of the preceding data bit.

Start each bit by outputting a tone burst:

```
TAPEO: MVI    B,9
T01:   MVI    A,0C0H
      CALL    BURST
```

Rotate register C through CY:

```
MOV    A,C
RAR
MOV    C,A
```

Move CY to the SOD enable bit position, A<sub>6</sub>. Simultaneously set A<sub>7</sub> to one, and clear all other bits. Output a tone burst or space, depending on the previous contents of CY:

```
MVI    A,01H
RAR
RAR
CALL    BURST
```

Clear the accumulator, and output a space:

```
XRA    A
CALL    BURST
```

Keep cycling until the full 9-bit sequence is finished:

```
DCR    B
JNZ    T01
RET
```

The BURST subroutine executes the SIM instruction CYCNO times, at intervals of 29 + 14 (HALFCYC) machine cycles. In between each SIM, bit A<sub>7</sub> is complemented. CYCNO should be an even number. If A<sub>6</sub> is set upon calling BURST a square-wave will be created. Otherwise, the same code sequence is followed but SOD does not change — thus a space results.

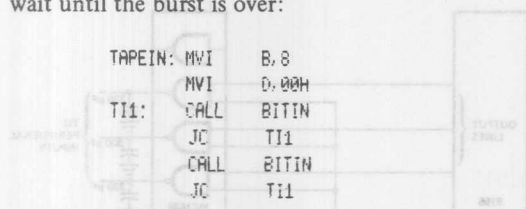
```
BURST: MVI    D,CYCNO    <7>
      BU1:   SIM
      MVI    E,HALFCYC  <7>
      BU2:   DCR    E      <4>
      JNZ    BU2        <7/10>
      XRI    80H        <7>
      DCR    D          <4>
      JNZ    BU1        <7/10>
      RET              <10>
```

#### INPUT ROUTINE

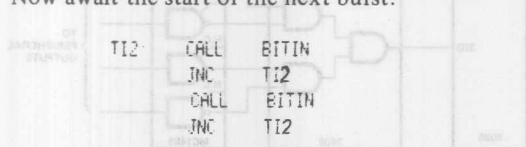
TAPEIN uses a subroutine called BITIN to move the data at the SID pin into the CY. The maximum rate at which SID is read is limited by a delay loop in BITIN.

Initialize the bit counter and the register D, which will keep track of the tone burst time. If a tone

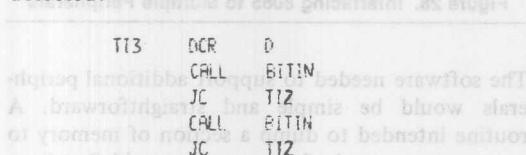
burst is being received when TAPEIN is called, wait until the burst is over:



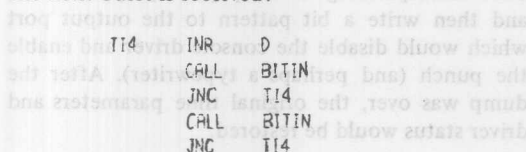
(Throughout this subroutine, a level transition is recognized only after it has been read once initially and then verified on the next reading. This provides some degree of software noise immunity.) Now await the start of the next burst:



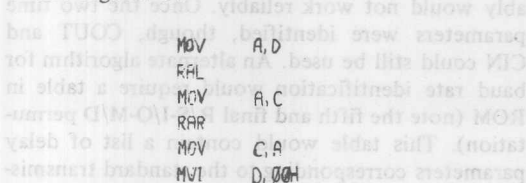
The next burst has now arrived. Keep reading the SID pin, decrementing register D (thus making it more negative), each cycle until the pause is detected:



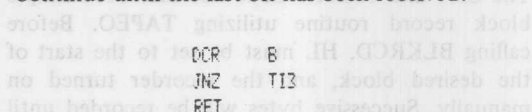
Now continue reading the SID pin, incrementing the D register (back towards zero), each cycle until the next burst is received:



Now, if the burst lasted longer than the space, D was not incremented all the way back to zero; it is still negative. If the space was longer, D was incremented up through zero; it is now positive. In other words, the sign bit of D will now correspond to the data bit that would lead to each of these results. Move the sign bit into the CY, then rotate it into register C:

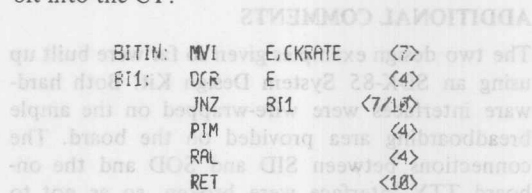


Continue until the last bit has been received:



(Notice that the first half of this subroutine is incorporated in the second half. In fact, the assembled listing included in the Appendix makes use of this fact to eliminate 24 bytes of duplicated code.)

BITIN waits a short time in order to regulate the sampling rate, then reads SID and moves the data bit into the CY:



The tone burst frequency and duration, and the TAPEIN sampling rate are determined by HALFCYC, CYCNO, and CKRATE. Tables 10 and 11 give typical values.

Table 10  
EXAMPLE COMBINATIONS OF HALFCYC AND CYCNO.  
ALL VALUES IN DECIMAL

APPROXIMATE TONE FREQUENCY	CORRESPONDING HALFCYC VALUE	RESULTING DATA RATE			
		8 4	20 10	100 50	CYCNO CYC/BURST
500 Hz	217	42	17	3.3	bps
1 kHz	108	83	33	6.6	bps
2 kHz	53	166	66	13	bps
5 kHz	20	414	166	33	bps
10 kHz	9	826	330	66	bps

Table 11  
MAXIMUM SAMPLING RATES  
FOR VARIOUS VALUES OF  
CKRATE

CKRATE VALUE	SAMPLING RATE (INCLUDING CALL & RET)
1	17.6 $\mu$ sec
20	104 $\mu$ sec
80	378 $\mu$ sec
250	1.14 msec

The CRT and Cassette Code also includes a simple block record routine utilizing TAPEO. Before calling BLKRCO, HL must be set to the start of the desired block, and the recorder turned on manually. Successive bytes will be recorded until the end of that page, i.e., until L is incremented to zero. The playback routine requires presetting HL to the target address and turning on the recorder before PLAYBK is called. These routines incorporate a long tone burst before each data block to allow a recorder with Automatic Gain Control to stabilize before the data starts.

### ADDITIONAL COMMENTS

The two design examples given so far were built up using an SDK-85 System Design Kit. Both hardware interfaces were wire-wrapped on the ample breadboarding area provided on the board. The connections between SID and SOD and the on-board TTY interface were broken, so as not to affect the 8085 I/O electrical characteristics.

The CRT interface was tested with a Beehive Mini-Bee II Terminal in the full duplex mode at each of its 14 possible transmission rates, from 110 to 9600 baud. It was also checked out at 19200 baud using a Beehive B-100 terminal. In addition, the software was exercised using an SBC 80/20 system as a variable baud rate character generator and receiver.

An additional advantage to having software selectable communications rates is that it would be possible to communicate with several system peripherals, each at its own preferred rate, without having to duplicate hardware. For example, the addition of a single 7408 AND gate and an output port (such as on the 8155) would make it possible to use the same two RS-232 circuits to interface with up to seven I/O devices (see Figure 28). Three of the MC1488 drivers have Enable inputs which can be controlled by the output port. One AND gate can be used to buffer the SOD line and drive the MC1488 Data inputs. The rest of the 7408 can be configured as a four input AND gate. This would act as an inverted logic OR gate to reduce the four MC1489 receiver outputs to a single line, which could be read by the SID. This assumes that only one input device (CRT, PTR) at a time will be used (which is usually the case in a non-time shared, interactive application), and that the unused devices are transmitting a logic one level (which should also be the case).

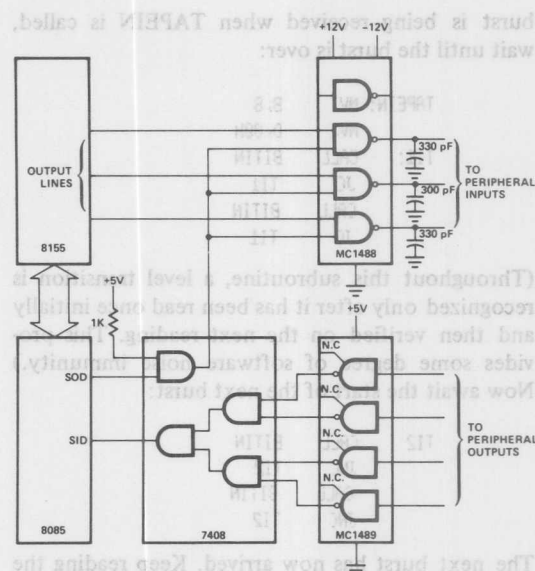


Figure 28. Interfacing 8085 to Multiple Peripherals

The software needed to support additional peripherals would be simple and straightforward. A routine intended to dump a section of memory to a paper tape punch, for example, would first have to store BITTIME and HALFBIT somewhere (perhaps on stack), load the variables with new parameters corresponding to the paper tape punch rate, and then write a bit pattern to the output port which would disable the console driver and enable the punch (and perhaps a typewriter). After the dump was over, the original time parameters and driver status would be restored.

As explained before, the BRID routine computed rate parameters based on the fact that an ASCII "space" character resulted in a zero level 6 bits long. Conceivably, some obscure peripherals might produce a transient between successive zero bits. (This might be the case, for example, if the signal was produced by mechanical rather than electronic means.) If so, the BRID algorithm used here probably would not work reliably. Once the two time parameters were identified, though, COUT and CIN could still be used. An alternate algorithm for baud rate identification would require a table in ROM (note the fifth and final R/S-I/O-M/D permutation). This table would contain a list of delay parameters corresponding to the standard transmis-



selected crystal components

components

were written for

## Temperature Sensor Code

ASM88 : F1 TEST SRC MOD85

1515-11 8888/8885 MACRO ASSEMBLER, V2.0

MODULE PAGE 1

LOC	OBJ	SEQ	SOURCE	STATEMENT
		1 ;		
		2 ;		
026C		3	MOVS	026CH ;EXPAND HEX TO DISPLAY, SDK MONITOR ROUTINE
0287		4	OUTPUT	0287H ;OUTPUT TO DISPLAY, SDK MONITOR ROUTINE
05F1		5	DELAY	05F1H ;DELAY DISPLAY, SDK MONITOR ROUTINE
		6 ;		
2000		7	ORG	2000H
		8 ;		
		9 ;		
		10 ;		
		11 ;		
2000	31C820	12	LXI	SP,20C8H ;INITIALIZE STACKPOINTER
2003	F3	13	DI	;DISABLE INTERRUPTS
		14 ;		
		15 ;		INITIALIZE COUNTER IN 8155 FOR COUNTDOWN MODE. LOAD COUNTER
		16 ;		WITH HIGHEST VALUE (3FFF).
		17 ;		
2004	3EBF	18	MVI	A,0BFH
2006	D325	19	OUT	25H ;ADDRESS FOR TOP HALF OF COUNTER
2008	3EFF	20	MVI	A,0FFH
200A	D324	21	OUT	24H ; " " LOWER HALF OF COUNTER
200C	3EC0	22	MVI	A,0C0H
200E	D320	23	OUT	20H ;COUNT DOWN MODE START
		24 ;		
		25 ;		PULSE THE ONE SHOT WITH A POSITIVE GOING PULSE ON THE SOD
		26 ;		OUTPUT PIN OF THE 8885.
		27 ;		
2010	3EC8	28	MVI	A,0C8H
2012	30	29	SIM	;OUTPUT A HIGH ON SOD LINE
2013	3E48	30	MVI	A,48H
2015	30	31	SIM	;OUTPUT A LOW ON SOD LINE
2016	FB	32	EI	;ENABLE INTERRUPTS(AFTER PULSE)
		33 ;		
		34 ;		IDLE UNTIL ONESHOT INTERRUPTS THE RST 6.5 PIN ON THE 8885
		35 ;		
2017	00	36	NPO	NOP
2018	C31720	37	JMP	NPO ; IDLE UNTIL INTERRUPT
		38 ;		
		39 ;		AFTER INTERRUPT, STOP COUNTER AND READ IN FINAL COUNT FROM
		40 ;		8155. STORE IN REGISTER PAIR BC.
		41 ;		
2018	3E40	42	CMVI	A,40H
201D	D320	43	OUT	20H ;STOP COUNTER
201F	0B24	44	IN	24H
2021	4F	45	MOV	C,A ;STORE LOWER ORDER BYTE IN C
2022	0B25	46	IN	25H
2024	47	47	MOV	B,A ;STORE HIGHER ORDER BYTE IN B
2025	263F	48	MVI	H,3FH ;LOAD HL WITH FULL START COUNT
2027	2EFF	49	MVI	L,0FFH
		50 ;		
		51 ;		ADJUST THE COUNT VALUE IN REGISTER BC TO REPRESENT ACTUAL
		52 ;		COUNT (SEE TEXT FOR EXPLANATION)

## Temperature Sensor Code (Cont'd)

1515-11 8888/8885 MACRO ASSEMBLER, V2.0

MODULE PAGE 2

LOC	OBJ	SEQ	SOURCE STATEMENT
		53 ;	
2029	CD6820	54	CALL ADJUST ; CONVERTS 8155 COUNT TO ACTUAL COUNT
		55 ;	
		56 ;	SETUP INITIALIZATION FOR SEARCH ROUTINE. ROUTINE LOOKS FOR TEMPERATURE
		57 ;	RANGE OF COUNT (SEE TEXT). SEARCH ONLY FOR UPPER HALF TO SIMPLIFY CODE.
		58 ;	
202C	2E80	59	MVI L, 80H ; SET HL TO BEGINNING OF SEARCH
202E	2620	60	MVI H, 20H ; STRING IN MEMORY
2030	B0	61	ORA B ; CLEAR CARRY FOR ROUTINE
2031	78	62	MOV A, B ; PLACE B INTO ACCUMULATOR
2032	0E01	63	MVI C, 1H ; SET TIMES THROUGH SEARCH
2034	CD9220	64	CALL SEARCH ; LOOKS FOR TEMP RANGE COUNT IS IN
		65 ;	
		66 ;	CHECK IF SEARCH WAS SUCCESSFUL. IF NOT THEN OUTSIDE ACCEPTABLE
		67 ;	RANGE.
		68 ;	
2037	3E80	69	MVI A, 80H ; DID L FIND LESS THAN AT
2039	AD	70	XRA L ; AT BEGINNING OF STRING?
203A	C9AF20	71	JZ TLOW ; TEMP BELOW ALLOWED LIMITS, SET PORT A
203D	3E80	72	MVI A, 80H ; DID C GET DECREMENTED?
203F	B9	73	CMF C ; IF SO, SEARCH DID NOT FIND
2040	CAB820	74	JZ THIGH ; TEMP ABOVE LIMITS, SET PORT B
		75 ;	
		76 ;	SOFTWARE MAP THE MATCH TO A TEMPERATURE IN DEGREES C BY ADDING
		77 ;	10 TO SEARCH ADDRESS. PLACE TEMPERATURE IN REGISTER E.
		78 ;	
2043	3E80	79	MVI A, 0AH ; SHIFT HL BY 10 (SOFTWARE MAP)
2045	85	80	ADD L
2046	6F	81	MOV L, A
2047	5E	82	MOV E, H ; READ IN TEMPERATURE
		83 ;	
		84 ;	SET UP INITIALIZATION FOR DISPLAYING TEMPERATURE USING SDK
		85 ;	MONITOR ROUTINES. FIRST EXPAND DE REGISTER AND THEN DISPLAY
		86 ;	FOR DELAY PERIOD.
		87 ;	
2048	0600	88	MVI B, 00H ; CLEAR DOT AT ADDRESS FIELD
204A	CD6C82	89	CALL HXDSP ; CALL EXPAND
204D	3E80	90	MVI A, 00H
204F	CD8702	91	CALL OUTPUT ; OUTPUT TO SDK DISPLAY
2052	11FF00	92	LXI D, 0FFH ; SET DELAY PERIOD
2055	CDF105	93	CALL DELAY ; DISPLAY FOR DELAY PERIOD
2058	CF	94	RST 1 ; SOFTWARE RESTART
		95 ;	
		96 ;	SUBROUTINES
		97 ;	
20AF		98 ORG 20AFH	
		99 ;	
		100 ;	
20AF	3E03	101 TLOW	MVI A, 03H
20B1	D320	102	OUT 20H
20B3	3EFF	103	MVI A, 0FFH ; SET PORT A AS 1'S
20B5	D321	104	OUT 21H
20B7	CF	105	RST 1
		106 ;	
		107 ;	

LOC	OBJ	SEQ	SOURCE STATEMENT
2088	3E03	108	THIGH: MVI A, 03H
2089	D320	109	OUT 20H
2090	3EFF	110	MVI A, 0FFH ; SET PORT B AS 1'S
2091	D322	111	OUT 22H
2092	CF	112	RST 1
		113	.
		114	.
2092		115	ORG 2092H
		116	.
		117	.
2092	BE	118	SEARCH: CMP M
2093	D8	119	RC
2094	23	120	INX H ; ELSE INCREMENT POINTER
2095	BE	121	CMP M ; COMPARE 2ND BYTE
2096	D8	122	RC
2097	23	123	INX H
2098	BE	124	CMP M ; COMPARE 3RD BYTE
2099	D8	125	RC
2099	23	126	INX H
2098	BE	127	CMP M ; COMPARE 4TH BYTE
209C	D8	128	RC
2090	23	129	INX H
209E	BE	130	CMP M ; COMPARE 5TH BYTE
209F	D8	131	RC
20A0	23	132	INX H
20A1	BE	133	CMP M ; COMPARE 6TH BYTE
20A2	D8	134	RC
20A3	23	135	INX H
20A4	BE	136	CMP M ; COMPARE 7TH BYTE
20A5	D8	137	RC
20A6	23	138	INX H
20A7	BE	139	CMP M ; COMPARE 8TH BYTE
20A8	D8	140	RC
20A9	23	141	INX H
20AA	00	142	DOR C ; HAS ENTIRE BLOCK BEEN
20AB	C29220	143	JNZ SEARCH ; SEARCHED? IF SO SET NO
20AC	09	144	RET ; LESS THAN AND RETURN.
		145	.
		146	; RESTART 6 5 JUMP ADDRESS
		147	.
20CE		148	ORG 20CEH
		149	.
		150	.
20CE	C31B20	151	JMP CNTU
		152	.
		153	.
		154	.
		155	.
		156	.
		157	.
		158	SEARCH COMPARE DATA STRING (SEE TEXT)
		159	.
		160	.
20B0		161	ORG 20B0H
		162	.

# Temperature Sensor Code (Cont'd)

IS15-11 8888/8885 MACRO ASSEMBLER, V2.0

MODULE PAGE 4

```

LOC OBJ      SEQ      SOURCE STATEMENT
2080 35      163 ;
2081 36      164      DB      35H, 36H, 37H, 38H, 39H, 3AH, 3BH, 3CH
2082 37
2083 38
2084 39
2085 3A
2086 3B
2087 3C
2088          165 ;
2089          166 ; SOFTWARE MAP TO TEMPERATURE
2090          167 ;
2091          168 ORG      2088H
2092          169 ;
2093          170 ;
2094          171      DB      21H, 23H, 25H, 28H, 31H, 35H, 39H
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174 ORG      2060H
2175 ;
2176 ;
2177 ; SUBROUTINE ADJUST FOR COUNT IN 8155
2178 ;
2179 ADJUST: MOV      A, B      ; LOAD ACCUMULATOR WITH UPPER HALF
2180 ANI      3FH      ; RESET UPPER TWO BITS, CLEAR CARRY
2181 RAR      ; ROTATE RIGHT THROUGH CARRY
2182 MOV      B, A      ; STORE SHIFTED VALUE BACK IN B
2183 MOV      A, C      ; LOAD ACCUMULATOR WITH LOWER HALF
2184 RAR      ; ROTATE WITH CARRY RIGHT
2185 MOV      C, A      ; STORE SHIFTED VALUE IN C
2186 RNC      ; 1ST HALF OR SECOND? IF SECOND RETURN
2187 CMC      ; CLEAR CARRY
2188 MOV      A, H      ; OBTAIN ONE HALF OF FULL COUNT
2189 RAR      ; IF HL IS 000 THIS CONTAINS
2190 MOV      H, A      ; ONE HALF (FULL COUNT-1), WHICH
2191 MOV      A, L      ; IS CORRECT
2192 RAR      ;
2193 MOV      L, A      ;
2194 DAD      B      ; DOUBLE PRECISION ADD
2195 MOV      B, H      ; RESTORE BC REGISTERS WITH COUNT
2196 MOV      C, L      ;
2197 RET
2198 ;
2199 ;
2200      END

```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

IS15-11 8888/8885 MACRO ASSEMBLER, V2.0

MODULE PAGE 5

ADJUST A 2060 CNTU A 2018 DELAY A 05F1 HX/SP A 026C NFO A 2017 OUTPUT A 0267 SEARCH A 2092  
THIGH A 2068 TLOW A 206F

ASSEMBLY COMPLETE, NO ERRORS



### CRT and Cassette Code

1515-II 3080/3085 ASSEMBLER. 01 0

1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26

PAGE 1

LOC 09.1

SEO

SOURCE STATEMENT

0 14 MO085 TITLE(0885 SERIAL 1/0 NOTE APPENDIX)

# **CRT and Cassette Code (Cont'd)**

IS15-II 8080/8085 ASSEMBLER, V1 0  
8085 SERIAL I/O NOTE APPENDIX

MODULE

PAGE 2

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	
		2 :	THE FOLLOWING PROGRAMS AND SUBROUTINES ARE DESCRIBED IN DETAIL
		3 :	IN INTEL CORPORATION'S APPLICATION NOTE AP-29, "USING THE 8085
		4 :	SERIAL I/O LINES". THE FIRST SECTION IS A GENERAL PURPOSE CRT
		5 :	INTERFACE WITH AUTOMATIC BAUD RATE IDENTIFICATION; THE SECOND
		6 :	SECTION IS A MAGNETIC TAPE INTERFACE FOR STORING DATA ON CASSETTE
		7 :	TAPE. THE CODE PRESENTED HERE IS ORIGINATED AT LOCATION 800H.
		8 :	AND MIGHT BE PART OF AN EXPANSION PROM IN AN INTEL SDK-85
		9 :	SYSTEM DESIGN KIT.
		10 :	
		11	
		12	
20C8		13 BITTIME EQU	20C8H ; ADDRESS OF STORAGE FOR COMPUTED BIT DELAY
20CA		14 HALFBIT EQU	20CAH ; ADDRESS OF STORAGE FOR HALF BIT DELAY
0008		15 BITS0 EQU	11 ; DATA BITS PUT OUT (INCLUDING TWO STOP BITS)
0009		16 BITS1 EQU	9 ; DATA BITS TO BE RECIEVED (INCLUDING ONE STOP BIT)
		17	
0000		18 ORG	800H ; STARTING ADDRESS OF SDK-85 EXPANSION PROM
		19	
		20 :	CRTTST CRT INTERFACE TEST. WHEN CALLED, AWAITS THE SPACE BAR BEING PRESSED ON
		21 :	THE SYSTEM CONSOLE, AND THEN RESPONDS WITH A DATA RATE VERIFICATION
		22 :	MESSAGE THERE AFTER, CHARACTERS TYPED ON THE KEYBOARD ARE ECHOED
		23 :	ON THE DISPLAY TUBE. WHEN A BREAK KEY IS TYPED, THE ROUTINE IS
		24 :	RE-STARTED, ALLOWING A DIFFERENT BAUD RATE TO BE SELECTED ON THE CRT.
0000 310020		25 CRTTST LXI	SP,20C0H
0003 3EC0		26 CRT1 MVI	A,0C0H ; SOD MUST BE HIGH BETWEEN CHARACTERS
0005 30		27 SIM	
0006 CD1A08		28 CALL	BRID ; IDENTIFY DATA RATE USED BY TERMINAL
0009 CD4708		29 CALL	SIGNON ; OUTPUT SIGNON MESSAGE AT RATE DETECTED
000C CD8A08		30 ECHO CALL	CIN ; READ NEXT KEYSTROKE INTO REGISTER C
000F 79		31 MOV	A,C
0010 B7		32 ORA	A ; CHECK IF CHARACTER WAS A (BREAK) (ASCII 00H)
0011 CA0308		33 JZ	CRT1 ; IF SO, RE-IDENTIFY DATA RATE
		34	; THIS ALLOWS ANOTHER RATE TO BE SELECTED ON CRT
0014 CD6908		35 CALL	COUT ; OTHERWISE COPY REGISTER C TO THE SCREEN
0017 C30C08		36 JMP	ECHO ; CONTINUE INDEFINITELY (UNTIL BREAK)
		37	
		38 :	BRID BAUD RATE IDENTIFICATION SUBROUTINE
		39 :	EXPECTS A (CR) (ASCII 20H) TO BE RECIEVED FROM THE CONSOLE.
		40 :	THE LENGTH OF THE INITIAL ZERO LEVEL (SIX BITS WIDE) IS MEASURED
		41 :	IN ORDER TO DETERMINE THE DATA RATE FOR FUTURE COMMUNICATIONS.
001A 20		42 BRID PIM	; VERIFY THAT THE "ONE" LEVEL HAS BEEN ESTABLISHED
001B B7		43 ORA	A ; AS THE CRT IS POWERING UP
001C F21A08		44 JP	BRID
001F 20		45 BR11 RIM	; MONITOR SID LINE STATUS
0020 B7		46 ORA	A
0021 FA1F08		47 JM	BR11 ; LOOP UNTIL START BIT IS RECIEVED
0024 21FAFF		48 LXI	H,-6 ; BIAS COUNTER USED IN DETERMINING ZERO DURATION
0027 1E04		49 BR13 MVI	E,04H
0029 10		50 BR14 DCR	E ; 52 MACHINE CYCLE DELAY LOOP
002A C22908		51 JNZ	BR14
002D 23		52 INX	H ; INCREMENT COUNTER EVERY 84 CYCLES WHILE SID IS LOW
002E 20		53 PIM	

LOC	OBJ	SEQ	SOURCE STATEMENT
002F	B7	54	ORA A
0030	F22700	55	JP BR13
		56	:CHL: NOW CORRESPONDS TO INCOMING DATA RATE
0033	E5	57	PUSH H
0034	24	58	INR H
0035	2C	59	INR L
0036	220820	60	SHLD BITTIME
0039	E1	61	POP H
003A	B7	62	ORA A
003B	7C	63	MOV A,H
003C	1F	64	RAR
003D	67	65	MOV H,A
003E	7D	66	MOV A,L
003F	1F	67	RAR
0040	6F	68	MOV L,A
0041	24	69	INR H
0042	2C	70	INR L
0043	220A20	71	SHLD HALFBIT
0046	C9	72	PET
		73	
		74	:SIGNON WRITES A SIGN-ON MESSAGE TO THE CRT AT WHAT SHOULD BE THE CORRECT RATE.
		75	: IF THE MESSAGE IS UNINTELLIGIBLE. WELL, SO IT GOES.
0047	215500	76	:SIGNON: LVI H,STRNG
004A	4E	77	:S1: MOV C,M
004B	AF	78	XRA A
004C	B1	79	ORA C
004D	C8	80	RZ
004E	CD6900	81	CALL COUT
0051	23	82	INX H
0052	C34A00	83	JMP S1
		84	
0055	00	85	:STRNG: DB 00H,0AH
0056	0A		:<CR><LF>
0057	42415544	86	DB 'BAUD RATE CHECK'
005B	20524154		
005F	45204348		
0063	454348		
0065	00	87	DB 00H,0AH
0067	0A		:<CR><LF>
0068	00	88	DB 00H
		89	:END-OF-STRING ESCAPE CODE
		90	:COUT: CONSOLE OUTPUT SUBROUTINE
		91	: WRITES THE CONTENTS OF THE C REGISTER TO THE CRT DISPLAY SCREEN
0069	F3	92	DI
006A	C5	93	PUSH B
006B	E5	94	PUSH H
006C	0600	95	MVI B,BIT50
006E	AF	96	XRA A
006F	3E00	97	MVI A,80H
0071	1F	98	RAR
0072	30	99	SIM
0073	2A0820	100	LHLD BITTIME
0076	20	101	DJR L

# **CRT and Cassette Code (Cont'd)**

IS15-II 8080/8085 ASSEMBLER, V1.8

MODULE

PAGE 4

8085 SERIAL I/O NOTE APPENDIX

LOC	OBJ	SEQ	SOURCE STATEMENT
0877	C27608	102	JNZ C02
087A	25	103	DCR H
087B	C27608	104	JNZ C02
087E	37	105	STC ;SET WHAT WILL EVENTUALLY BECOME A STOP BIT
087F	79	106	MOV A,C ;ROTATE CHARACTER RIGHT ONE BIT,
0880	1F	107	RAR ;\ MOVING NEXT DATA BIT INTO CARRY
0881	4F	108	MOV C,A
0882	05	109	DCR B ;CHECK IF CHARACTER (AND STOP BIT(S)) DONE
0883	C26F08	110	JNZ C01 ;IF NOT, OUTPUT CURRENT CARRY
0886	E1	111	POP H ;RESTORE STATUS AND RETURN
0887	C1	112	POP B
0888	FB	113	EI
0889	C9	114	RET
		115	
		116	:CIN CONSOL INPUT SUBROUTINE WAITS FOR A KEYSTROKE AND
		117	RETURNS WITH 8 BITS IN REG C.
088A	F3	118	CIN: DI
088B	E5	119	PUSH H
088C	0E09	120	MVI B,BITSI ;DATA BITS TO BE READ (LAST RETURNED IN CY)
088E	20	121	CI1: RIM ;WAIT FOR SYNC BIT TRANSITION
088F	B7	122	ORA A
0890	F8E08	123	JM CI1
0893	2AC920	124	LHLD HALFBIT
0896	20	125	DCR L ;WAIT UNTIL MIDDLE OF START BIT
0897	C29608	126	JNZ CI2
089A	25	127	DCR H
089B	C29608	128	JNZ CI2
089E	2AC920	129	CI2: LHLD BITTIME ;WAIT OUT BIT TIME
08A1	20	130	DCR L
08A2	C2A108	131	JNZ CI4
08A5	25	132	DCR H
08A6	C2A108	133	JNZ CI4
08A9	20	134	RIM ;CHECK SID LINE LEVEL
08AA	17	135	RAL ;DATA BIT IN CY
08AB	05	136	DCR B ;DETERMINE IF THIS IS FIRST STOP BIT
08AC	CABE08	137	JZ CI5 ;IF SO, JUMP OUT OF LOOP
08AF	79	138	MOV A,C ;ELSE ROTATE INTO PARTIAL CHARACTER IN C
08B0	1F	139	RAR ;ACC HOLDS UPDATED CHARACTER
08B1	4F	140	MOV C,A
08B2	00	141	NOF ;EQUALIZES COUT AND CIN LOOP TIMES
08B3	C29E08	142	JMP CI3
08B6	E1	143	CI5: POP H
08B7	FB	144	EI
08B8	C9	145	RET ;CHARACTER COMPLETE
		146	
		147	:*****
		148	
		149	: THE FOLLOWING CODE IS USED BY THE CASSETTE INTERFACE.
		150	SUBROUTINES TAPED AND TAPEIN ARE USED RESPECTIVELY
		151	TO OUTPUT OR RECEIVE AN EIGHT BIT BYTE OF DATA. REGISTER C
		152	HOLDS THE DATA IN EITHER CASE. REGISTERS A,B,&C ARE ALL DESTROYED.
0810		153	CYOND EQU 16 ;TWICE THE NUMBER OF CYCLES PER TONE BURST
081E		154	HALFCY EQU 20 ;DETERMINES TONE FREQUENCY

# CRT and Cassette Code (Cont'd)

TS15-II 8080/8085 ASSEMBLER, V1 0  
8085 SERIAL I/O NOTE APPENDIX

MODULE

PAGE 5

LOC	OBJ	SEQ	SOURCE STATEMENT
0016		155	CHRATE EQU 22 ;SETS SAMPLE RATE
00FA		156	LEADER EQU 250 ;NUMBER OF SUCCESSIVE TONE BURSTS COMPRISING LEADER
00FA		157	LOPCHK EQU 250 ;USED IN PLAYEK TO VERIFY PRESENCE OF LEADER
		158	
		159	BURPCD ;OUTPUTS A VERY LONG TONE BURST ((LEADER) TIMES
		160	THE NORMAL BURST DURATION) TO ALLOW RECORDER ELECTRONICS
		161	AND AGC TO STABILIZE. THEN OUTPUTS THE REMAINDER OF THE
		162	256 BYTE PAGE POINTED TO BY (H), STARTING AT BYTE (L).
00B9 0EFA		163	BURPCD MVI C,LEADER;SET UP LEADER BURST LENGTH
00BB 3EC0		164	MVI A,000H ;SET ACCUMULATOR TO RESULT IN TONE BURST
00BD CDF000		165	BP1 CALL BURST ;OUTPUT TONE
00CD 00		166	DCR C
00C1 C2B003		167	JNZ BP1 ;SUSTAIN LEADER TONE
00C4 AF		168	XRA A ;CLEAR ACCUMULATOR & OUTPUT SPACE, SO THAT
00C5 CDF000		169	CALL BURST ;START OF FIRST DATA BYTE CAN BE DETECTED
00C8 4E		170	BP2 MOV C,H ;GET DATA BYTE TO BE RECORDED
00C9 C0C100		171	CALL TAPEO ;OUTPUT REGISTER C TO RECORDER
00CC 2C		172	INR L ;POINT TO NEXT BYTE
00CD C2C000		173	JNZ BP2
00D0 C9		174	RET ;AFTER BLOCK IS COMPLETE
		175	
		176	
		177	TAPEO ;OUTPUTS THE BYTE IN REGISTER C TO THE RECORDER.
		178	REGISTERS A,B,C,D,&E ARE ALL USED.
00D1 F3		179	TAPEO: DI
00D2 05		180	PUSH D ;D&E USED AS COUNTERS BY SUBROUTINE BURST
00D3 0609		181	MVI E,9 ;WILL RESULT IN 8 DATA BITS AND ONE STOP BIT
00D5 AF		182	XRA A ;CLEAR ACCUMULATOR
00D6 2EC0		183	MVI A,000H ;SET ACCUMULATOR TO CAUSE A TONE BURST
00D8 CDF000		184	CALL BURST
00DB 79		185	MOV A,C ;MOVE NEXT DATA BIT INTO THE CARRY
00DD 1F		186	RAR
00DD 4F		187	MOV C,A ;CARRY WILL BECOME SOD ENABLE IN BURST ROUTINE
00DE 3E01		188	MVI A,01H ;SET BIT TO BE REPEATEDLY COMPLEMENTED IN BURST
00E0 1F		189	RAR
00E1 1F		190	RAR
00E2 CDF000		191	CALL BURST ;OUTPUT EITHER A TONE OR A PAUSE
00E5 AF		192	XRA A ;CLEAR ACCUMULATOR
00E6 CDF000		193	CALL BURST ;OUTPUT PAUSE
00E9 05		194	DCR B
00EA C2D500		195	JNZ T01 ;REPEAT UNTIL BYTE FINISHED
00ED D1		196	POP D ;RESTORE STATUS AND RETURN
00EE FB		197	EI
00EF C9		198	RET
		199	
00F0 1610		200	BURST: MVI D,CYCNO ;SET NUMBER OF CYCLES
00F2 30		201	BUI1 SIM ;COMPLEMENT SOD LINE IF SOD ENABLE BIT SET
00F3 1E1E		202	MVI E,HALFCYC
00F5 1D		203	BUI2 DCR E ;REGULATE TONE FREQUENCY
00F6 C2F500		204	JNZ BUI2
00F9 EE00		205	XRI 00H ;COMPLEMENT SOD DATA BIT IN ACCUMULATOR
00FB 15		206	DCR D
00FC C2F200		207	JNZ BUI1 ;CONTINUE UNTIL BURST (OR EQUIVALENT PAUSE) FINISHED



# CRT and Cassette Code (Cont'd)

IS15-II 8080/8085 ASSEMBLER, V1.0  
8085 SERIAL I/O NOTE APPENDIX

MODULE

3.0000

PAGE

6

00000000 0000 0000 11-1111  
00000000 0000 0000 11-1111

LOC OBJ SEQ SOURCE STATEMENT

08FF C9 208 RET

209

210 ;PLAYBK

WAITS FOR THE LONG LEADER BURST TO ARRIVE, THEN CONTINUES

0899 A 1A8 0899 A 03A4 211 ; 8085 A MITT19 READING BYTES FROM THE RECORDER AND STORING THEM

2789 A 508 5789 A 212 ; 8126 A 0189 IN MEMORY STARTING AT LOCATION <HL>

0800 A 013 0800 A 213 ; 1A00 A CONTINUES UNTIL THE END OF THE CURRENT PAGE (<LD>=0FFH) IS REACHED

0189 A 0000 0EFA 214 PLAYBK: MVI C,LDROCHK ;<LDROCHK> SUCCESSIVE HIGHS MUST BE READ

0809 A 0902 CD3D09 215 PB1: CALL BITIN ; TO VERIFY THAT THE LEADER IS PRESENT

3189 A 0905 D20009 216 2189 A JNC PLAYBK ; AND ELECTRONICS HAS STABILIZED

0908 00 217 DCR C

0909 C20209 218 JNZ PB1

090C CD1509 219 PB2: CALL TAPEIN ;GET DATA BYTE FROM RECORDER

090F 71 220 MOV M,C ;STORE IN MEMORY

0910 2C 221 INR L ;INCREMENT POINTER

0911 C20C09 222 JNZ PB2 ;REPEAT FOR REST OF CURRENT PAGE

0914 C9 223 RET

224

225 ;TAPEIN CASSETTE TAPE INPUT SUBROUTINE. READS ONE BYTE OF DATA

226 ; FROM THE RECORDER INTERFACE AND RETURNS WITH THE BYTE IN REGISTER C.

0915 0609 227 TAPEIN: MVI B,9 ;READ EIGHT DATA BITS

0917 1600 228 TI1: MVI D,00H ;CLEAR UP/DOWN COUNTER

0919 15 229 TI2: DCR D ;DECREMENT COUNTER EACH TIME ONE LEVEL IS READ

091A CD3D09 230 CALL BITIN

091D DA1909 231 JC TI2 ;REPEAT IF STILL AT ONE LEVEL

0920 CD3D09 232 CALL BITIN

0923 DA1909 233 JC TI2

0926 14 234 TI3: INR D ;INCREMENT COUNTER EACH TIME ZERO IS READ

0927 CD3D09 235 CALL BITIN

092A D22609 236 JNC TI3 ;REPEAT EACH TIME ZERO IS READ

092D CD3D09 237 CALL BITIN

0930 D22609 238 JNC TI3

0933 7A 239 MOV A,D

0934 17 240 RAL ;MOVE COUNTER MOST SIGNIFICANT BIT INTO CARRY

0935 79 241 MOV A,C

0936 1F 242 RAR ;MOVE DATA BIT RECEIVED (CY) INTO BYTE REGISTER

0937 4F 243 MOV C,A

0938 05 244 DCR B

0939 C21709 245 JNZ TI1 ;REPEAT UNTIL FULL BYTE ASSEMBLED

093C C9 246 RET

247

093D 1E16 248 BITIN: MVI E,CKRATE

093F 1D 249 BI1: DCR E

0940 C23F09 250 JNZ BI1 ;LIMIT INPUT SAMPLING RATE

0943 20 251 RIM ;SAMPLE SID LINE

0944 17 252 RAL ;MOVE DATA INTO CY BIT

0945 C9 253 RET

254

255 END

PUBLIC SYMBOLS

## EXTERNAL SYMBOLS

## USER SYMBOLS

B11	A 002F	BITIN	A 003D	BIT50	A 000B	BITTIM	A 20C8	BLKRCO	A 0089	BR1	A 008D
BR2	A 00C9	BP11	A 001F	BR12	A 0027	BR14	A 0029	BU1	A 00F2	BU2	A 00F5
BURST	A 00F0	C11	A 008E	C12	A 0096	C13	A 009E	C15	A 0086	CIN	A 008A
CKRATE	A 0016	C01	A 006F	C02	A 0076	COUT	A 0069	CRT1	A 0003	CRTTST	A 0000
ECHO	A 0000	HALFBI	A 20CA	HALFCY	A 001E	LDCHK	A 00FA	LEADER	A 00FA	PB1	A 0002
PLAYBK	A 0000	S1	A 004A	SIGNON	A 0047	STPNG	A 0055	TAPEIN	A 0915	TAPEO	A 00D1
TI2	A 0919	TI2	A 0925	T01	A 00D5					TI1	A 0917

ASSEMBLY COMPLETE. NO ERROR(S)

328: TAPEIN CASSETTE TAPE INPUT SUBROUTINE PERIOD ONE BYTE OF DATA  
 329: FROM THE RECORDER INTERFACE AND RETURNS WITH THE BYTE IN REGISTER C

330: READ EIGHT DATA BITS  
 331: WAIT  
 332: CLEAR UP/DOWN COUNTER  
 333: INCREMENT COUNTER EACH TIME ONE LEVEL IS READ

334: REPEAT IF STILL AT ONE LEVEL

335: INCREMENT COUNTER EACH TIME ZERO IS READ

336: REPEAT EACH TIME ZERO IS READ

337: MOVE COUNTER MOST SIGNIFICANT BIT INTO CARRY

338: MOVE DATA BIT RECEIVED (CY) INTO BYTE REGISTER

339: REPEAT UNTIL FULL BYTE ASSEMBLED

340: LIMIT INPUT SAMPLING RATE

341: SAMPLE 310 LINE

342: MOVE DATA INTO CY BIT

PUBLIC SYMBOLS

# CRT and Cassette Code (Cont'd)

IS15-II ASSEMBLER SYMBOL CROSS REFERENCE V1.0

PAGE 1

BIT	249#	250				
BITIN	215	230	222	235	227	246#
BITSI	16#	120				
BITSO	15#	95				
BITTIM	13#	60	100	129		
BLKROD	163#					
BR1	165#	167				
BR2	170#	173				
BRI1	45#	47				
BRI2	49#	55				
BRI4	50#	51				
BRID	28	42#	44			
BU1	201#	207				
BU2	203#	204				
BURST	165	169	184	191	193	200#
CI1	121#	123				
CI2	125#	126	128			
CI3	129#	142				
CI4	130#	131	133			
CI5	137	143#				
CIN	30	119#				
CKRATE	155#	248				
CO1	97#	110				
CO2	101#	102	104			
COUT	25	81	92#			
CRT1	26#	33				
CRTTST	25#					
CYCHO	153#	200				
ECHO	20#	36				
HALFBI	14#	71	124			
HALFCY	154#	202				
LDPCHK	157#	214				
LEADER	156#	163				
PE1	215#	218				
PE2	219#	222				
PLAYBK	214#	216				
S1	77#	83				
STGNON	29	76#				
STRNG	76	85#				
TAPEIN	219	227#				
TAPEO	171	179#				
TI1	228#	245				
TI2	229#	231	233			
TI3	234#	236	238			
TO1	182#	195				

\*POSS REFERENCE COMPLETE

